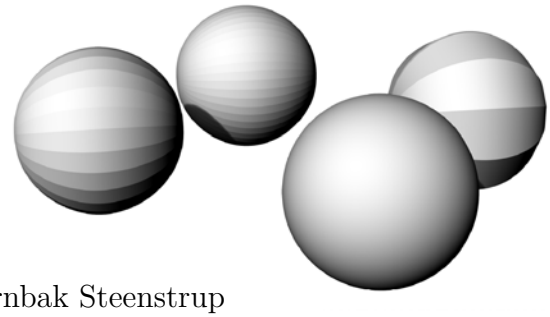
Rationalization with ruled surfaces in architecture

Steenstrup, Kasper Hornbak

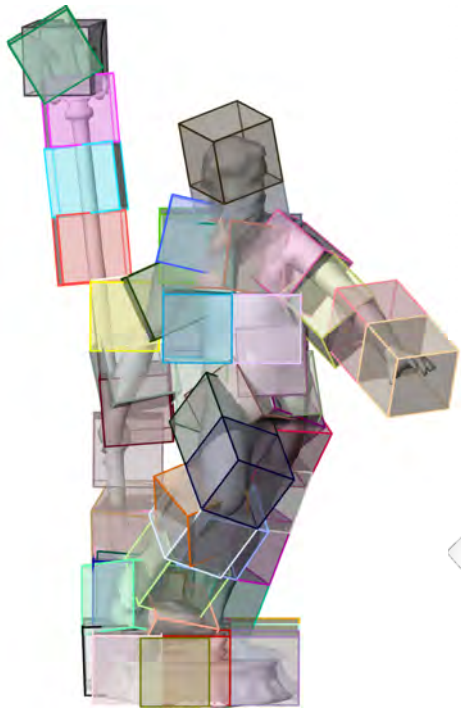
Publication date:
2016

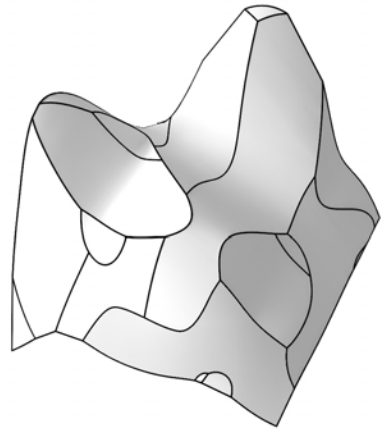
Document Version
Publisher's PDF, also known as Version of record

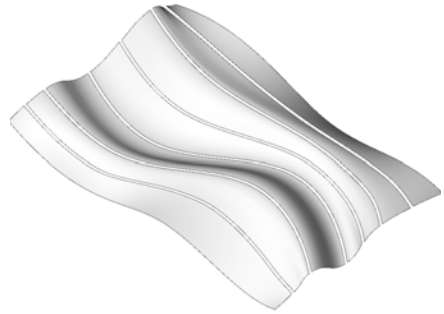
Rationalization with ruled surfaces in architecture



Kasper Hornbak Steenstrup
April 2016
PHD-2016-413







DTU Compute
Department of Applied Mathematics and Computer Science

Abstract

This thesis addresses the problems of rationalizing and segmenting large scale 3D models, and how to handle difficult production constraints in this area. The design choices when constructing large scale architecture are influenced by the budget. Therefore I strive to minimize the amount of time and material needed for production. This makes advanced free form architecture viable for low cost projects, allowing the architects to realize their designs.

By pre-cutting building blocks using hot wire robots, the amount of milling necessary can be reduced drastically. I do this by rationalizing the intended shape as a piecewise ruled surface; the developed method was able to cut away up to 95% of the excess material. Methods were developed to minimize the number of blocks necessary to build advanced large scale 3D shapes. Using stochastic optimization to guide the segmentation, it was possible to remove up to 48% of the building blocks. Hot blade cutting for constructing models with positive Gauss curvature is an upcoming technology. Three segmentation algorithms were developed to solve construction constraints that arises when using this technique. One of the algorithms focusses on creating an aesthetic segmentation.

Resume

Denne afhandling omhandler segmentering og approksimering af 3D modeller og de problemer der opstår når modellerne når en størrelse, hvor 3D print ikke er muligt. Ved opførelsen af store bygningsværker kan omkostningerne være en hindring for kreativiten. I denne afhandling forsøger jeg at minimere produktionstid samt mængden af byggematerialer, for at gøre det nemt og billigt at realisere arkitektoniske mesterværker.

Mængden af fræsning kan reduceres kraftigt ved at glødetrådsskære byggeklodserne forinden. Jeg har udviklet en metode til at approksimere overflader med stykvist retlinjede flader, og derved opnået at fjerne op til 95% af det overflødige materiale. Ved at bruge stokastisk optimering, til at segmentere 3D modeller, lykkedes det at reducere antallet af bygningsklodser med op til 48%. En ny og lovende teknologi er glødeklingeskæring, der kan udskære flader med positiv Gauss krumning. Vi har udviklet tre segmenteringsalgorithmer, som løser produktionsbegrænsninger, der opstår med denne teknik. En af disse algorithmer fokuserer på at lave den mest æstetiske segmentering.

Preface

This thesis presents the results of my research between the years 2013 and 2016. Throughout my PhD I have focused on segmentation and rationalization for the industry, in particular for large scale structures, and how to overcome the problems and constraints that arise when using different building techniques.

This thesis contains several papers and manuscripts that I authored or co-authored; Chapter 1 introduces the BladeRunner project, the chapters, and preliminaries. Chapter 2 is a paper [SNS+] describing pre-cutting of blocks and is submitted for the conference *AAG 2016*. Chapter 3 is a manuscript [SBSG] describing block segmentation and will be submitted to *SIGGRAPH Asia 2016*. Chapter 4 is an extension of what is introduced in the paper [BBaE+]. Chapter 5 describes three algorithms solving Elastica segmentation, and one of the algorithms can be found in the paper [SFN+16] from the conference Rob|Arch 2016.

Throughout my time as a PhD student, I have been grateful for the financial support from the *Innovation Fund Denmark* and the grant given to the BladeRunner project. The *Otto Mønsted* and *Augustinus* foundation have also helped me with economic support for travelling abroad, for which I am grateful. I would also like to thank the companies that I have worked closely with: The robotics company *Odico*, who have had the industrial and robotic know-how, the architect company *3XN*, which provided a lot of test cases and insight to problems in the architectural world, the *DTI* with knowledge of robots, the concrete company *CONFAC* for showing and producing the final product, the university UBC for letting me visit for five months, and the university *DTU* where I have conducted me PhD. Without all of this, my PhD would not have been the same.

I had the great opportunity to be part of three scientific groups: the *Mathematics* section and the *Image Analysis & Computer Graphics* section at DTU, and the department of *Computer Science* at *UBC*. I want to give a special thanks to my main supervisor Jens Gravesen who have not only helped me in my studies, but also done it with an expertise and a good spirit, I have not seen elsewhere. I will like to thank co-supervisor Andreas Bærentzen from DTU and Alla Sheffer from *UBC* for broadening my mind on geometry. I will also like to thank Toke Bjerge Nørbjerg, my fellow PhD student on the project. It has been a joy working with you all.

Last but not least I would like to thank my partner Irene L. T. Heilmann for

the support and for helping me throughout the PhD. From text correction and mathematics calculation to general support in what to do next, Irene have always been there.

Contents

List of floats

Tables

Figures

Chapter 1

Introduction

1.1 BladeRunner

The BladeRunner project is a three year research project funded by the Innovation Fund Denmark, and the participants are the three industrial firms Odico, 3XN, and CONFAC as well as the two research institutions Technical University of Denmark (DTU) and Danish Technological Institute (DTI). When architects design buildings with advanced curves, the construction process requires special casting molds. The architectural model goes through rationalization, in order for robots to cut the model from expanded polystyrene (EPS) blocks using a hot wire or a hot blade. The blocks are put together to form the casting mold and concrete is poured on top, finally creating the architect's design. When the architects' designs are more complex, the construction process becomes more complicated, and the cost of the building increases rapidly. The goal of the BladeRunner project is to explore and research the technology of hot wire and hot blade cutting in the industry. Hot wire cutting was from the start of the project used by Odico on a commercial level, whereas hot blade cutting was more of a theoretic concept.

The diversity of the partners in the BladeRunner project, means that most of the processes, from designing a model to casting it in concrete, are covered. The roles, of the different partners in the project, are:

- 3XN is an architect firm where case studies for the project arises.
- DTU Compute covers the rationalization and segmentation of the design.
- DTI develops the robot setup for the hot blade and (together with DTU Mechanical Engineering) develops the design of the hot blade.

- DTU Mechanical Engineering simulates blade expansion and material components, and (together with DTI) develops the design of the hot blade.
- Odico is a robotics company. They have a hot wire robot setup and handle coating of EPS blocks for fabrication.
- CONFAC is a concrete casting firm where the cut blocks are used for concrete casting.

A hot wire is a thin metal wire heated to approximately 200°C. The wire is held by a robot and cuts through EPS blocks by melting the material, it comes in contact with. The wire forms a straight line at all times, producing shapes of so-called ruled surfaces. After cutting, the blocks can be used as molds for concrete casting. The block can also be used directly as building material if given a coating that is durable. The robot setup and the result of cutting a block twice is seen in Figure 1.1.

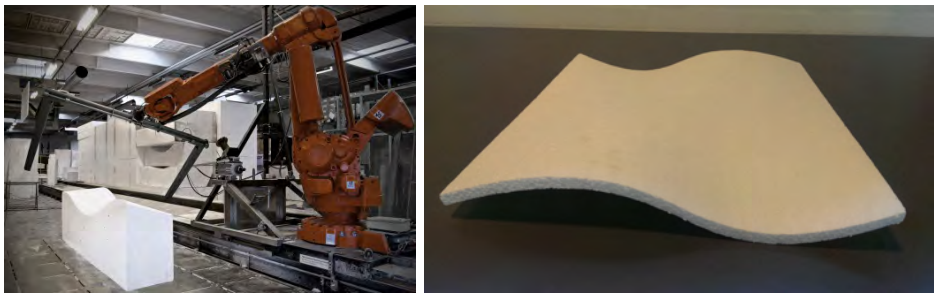


Figure 1.1: Left: An ABB robot setup with a hot wire mounted in the two ends of the robot's frame. Right: The result of an EPS block after two cuts. This model was created at the workshop *Research day 2015* at DTU

The hot blade is a metal blade heated to approximately 350°C mounted between two robot arms. By changing the positions and angles of the robot's hands, the blade bends in different shapes. In Figure 1.2 is shown the robot setup and the result of cutting the same block twice; notice this shape cannot be cut by a hot wire. The properties of the hot blade will be discussed further in chapter 5. Depending on the robotic setup, the blade can be exchanged with a tube.

The process of casting a model in concrete at CONFAC goes as follows. The blocks are tightly pack in a large rectangular frame with the cut side facing up. Then concrete is poured onto the blocks and at last the frame is shaken so the concrete settles into all the recesses in the mold. This means the shape of the blocks are the negative of the final design. Since multiple blocks are placed side by side, the seams between the blocks need to uphold casting tolerance standards. An illustration of twelve blocks collected in an frame, after they were cut with a hot wire, and the result of the concrete casting is seen in Figure 1.3.

At DTU Compute (Department of Applied Mathematics and Computer Science), we are working with the segmentation and rationalization of the model de-

Figure 1.2: Left: The hot blade setup at DTI. The blade is bent and the robot is holding the block, while feeding it to the blade. Right: An EPS block cut twice by a hot blade.

signs. The designs are given as computer aided design (CAD) models. Segmentation consists of dividing the model into regions that represents the EPS blocks. Depending on whether a hot wire or a hot blade is used, each region needs to be approximated with a ruled surface or a surface swept by a moving Euler elastica, respectively. The rationalization depends on the segmentation, and therefore it can be beneficial to consider the two tasks together. The regions should fit together smoothly and combined they should give satisfactory representation of the original surface. My contribution to the project consist of developing several algorithms for segmentation and for rationalization of ruled surfaces, during my PhD.

Figure 1.3: The casting at CONFAC. Left: Twelve EPS block in a frame being prepared for casting. Right: The result after casting. Observe how the result is the negative of the EPS mold.

From an industrial perspective the hot wire and hot blade technology are very interesting, since large scale structures are expensive to construct, especially when they include complicated geometry. In particular geometry with positive gauss curvature is expensive to manufacture, since the current technologies to create the form works are CNC-Milling, manual sculpturing, 3D printing, or bend/carved

wooden frames. All of these techniques are either slow, expensive or both, in contrast to using EPS blocks cut by a hot wire or hot blade, which is both fast, cheap and environmentally friendly.

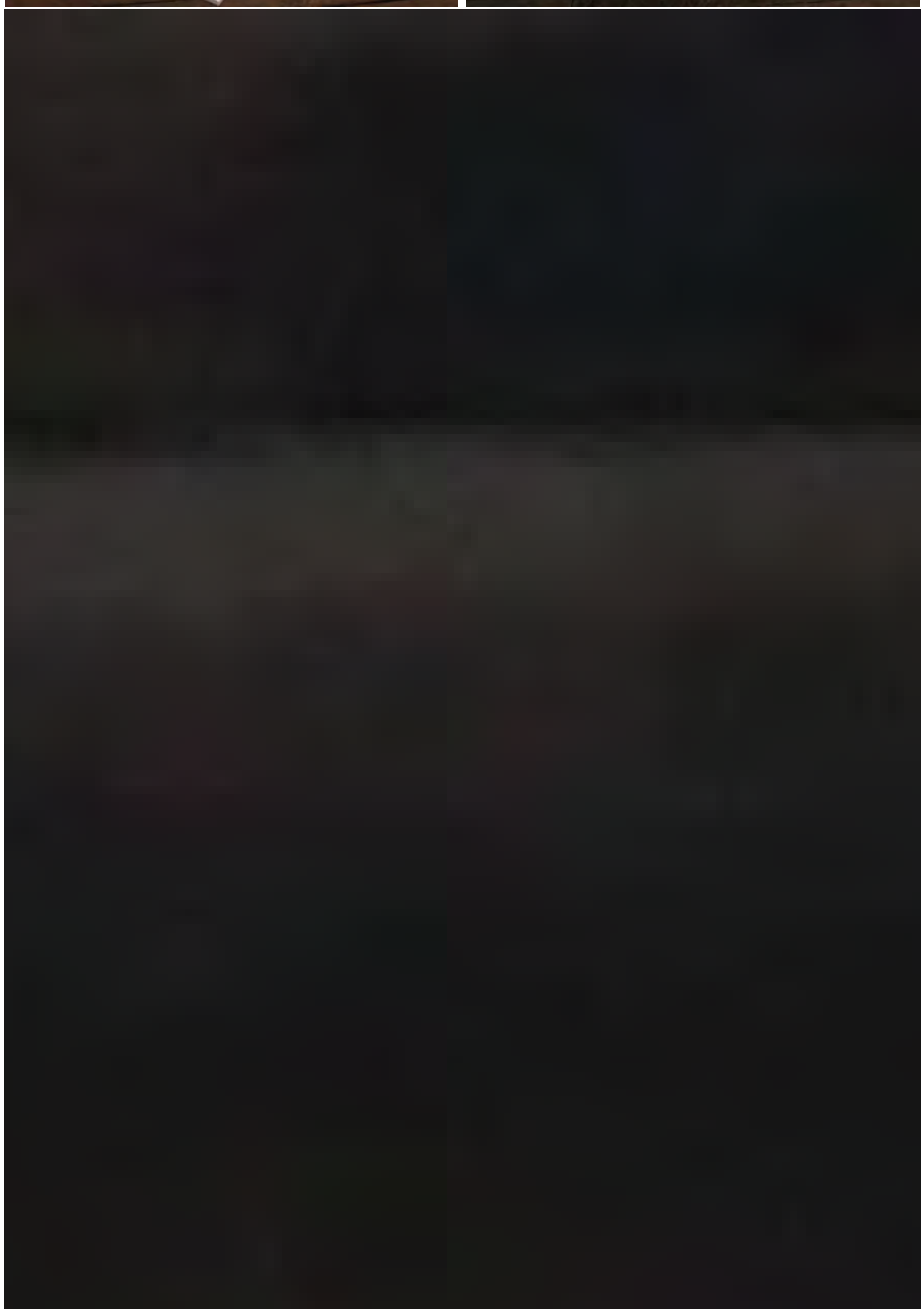
The hot wire technology is currently being used in industrial projects. An inspiring domicile is under construction in Vejle[1], where form works are cut by a hot wire at Odico. Other modern building projects where hot wire or hot blade techniques could have been used range from the roof of Nørrebro train station to the museum lobby at Louisiana, see Figure 1.4. The technology has the potential to not only lower the cost of high prestige buildings, but to also allow low budget projects to have much more intriguing designs. My personal ambition for the BladeRunner project, is the world will be a more beautiful place to live in.

Figure 1.4: Two construction projects where our technology would have been an excellent choice. Left: Nørrebro train station. Right: The Louisiana State Museum and Sports Hall of Fame.[2]

The hot blade technology is still in the research state, and its possibilities was presented by the BladeRunner team at the Rob|Arch 2016 conference with a paper [SFN+16] and a workshop for 18 architect students. The workshop's main contribution was a design tool allowing the students to design surfaces that by could be cut by a hot blade and a robot cutting the models. This setup gave a short feedback loop about the design tool. The students together with the BladeRunner team created several designs that were cut on site, see Figure 1.5. In the bottom part of the figure is the extraordinary result of students designing a model consisting of multiple blocks. The design tool, feedback from the students, and the work flow of the process, led to a submission in the *AAG2016* conference [BBC+].

[1] ing.dk/artikel/havnedomicil-giver-byggerobot-gennembrud-183314

[2] Nørrebro picture taken by *Inhabitat* under the licence https://creativecommons.org/licenses/by-nc-nd/2.0/, Louisiana picture taken by *Larry Rowland* under the licence https://creativecommons.org/licenses/by/2.0/



1.2 Chapters

Chapter 2 presents an optimization algorithm for pre-cutting of blocks using a hot wire. The method handles the demand for rapid production by removing as much material from a building block before milling the rest of the model. The model is rationalized by a piecewise ruled surface, with production constraints not solved elsewhere in the literature. The chapter is a paper submitted to the *AAG 2016* conference.

Chapter 3 presents an algorithm for volumetric segmentation of free form architecture models. The method handles the demand for saving construction material, by drastically lowering the number of building blocks needed to manufacture the models. The chapter is a manuscript that will be submitted to the conference *SIGGRAPH Asia 2016*.

Chapter 4 is an extension to my contribution to [BBaE+] and discusses the question of whether to segment or rationalize a model first. An algorithm is developed to tackle this question and automatically produce piece-wise ruled surface rationalization of a mesh. Such an algorithm is not seen in the literature so far, and the plan is to make a separate manuscript based on this algorithm.

Chapter 5 concerns the constraints involved in hot blade cutting and how to fulfill these constraints by segmentation. The method introduced in section 5.2 is my contribution to the paper [SFN+16], and is a greedy algorithm that solves fabrication constraints. Sections 5.3 and 5.4 presents collaborate work with Toke Bjerge Nørbjerg and describe two other methods for solving different constraints. The algorithm in section 5.3 lets the segmentation follow the geometry of model, whereas the algorithm in section 5.4 focusses on creating the largest possible segmentation regions. The work in these last two sections still needs some refinement before it is ready for publication.

1.3 Preliminaries

This section briefly introduces splines, ruled surfaces, and curvature, which are concepts used throughout this thesis. Ruled surfaces are also linked to hot wire cutting.

1.3.1 Splines

In the literature the word spline has many different meanings. In this thesis a spline is a piece-wise polynomial curve,

$$s_{\mathbf{u}}(v) = \sum_{i=1}^{n} \beta^p_{\mathbf{u},i}(v)\mathbf{c}_i \,, \tag{1.1}$$

where $\mathbf{c}$ is the set of n control points, and $\beta^p_{\mathbf{u},i}$ are B-splines on some given knot vector $\mathbf{u}$. The parameter where two polynomials meet are called the knots, and they form a non-decreasing sequence, $\mathbf{u} = [t_0, .., t_m]$. A knot is allowed to repeat,

and the number of repetitions is called the multiplicity, denoted μ. The degree of the B-spline is $p = m - n - 1$.

The B-spline $\beta^p_{\mathbf{u},i}$ can be derived from the Cox-de Boor recursion formula as

$$\beta^0_{\mathbf{u},i} = \begin{cases} 1 & \text{if } t_i \leq u < t_{i+1} \text{ and } t_i < t_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{1.2}$$

$$\beta^j_{\mathbf{u},i} = \frac{u - t_i}{t_{i+j} - t_i}\beta^{j-1}_{\mathbf{u},i} + \frac{t_{i+j+1} - u}{t_{i+j+1} - t_{i+1}}\beta^{j-1}_{\mathbf{u},i+1}\,, \tag{1.3}$$

where $j = 1, .., p$. With this representation, a B-spline have $C^{p-\nu}$ continuity at a knot with multiplicity ν. This is seen in Figure 1.6, where the knot vector is indicated by the small triangles in the bottom. Another property of splines, is that they are easy to differentiate, and differentiating a spline of degree p gives another spline of degree p-1.

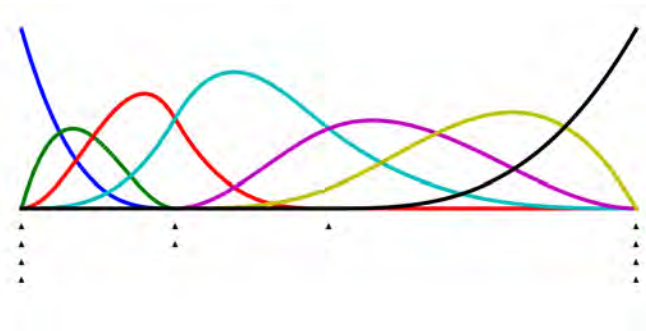


Figure 1.6: The basis functions $\beta^3_{\mathbf{u},i}$ having the knot vector $\mathbf{u} = [0, 0, 0, 0, 0.25, 0.25, 0.5, 1, 1, 1, 1]$. Since the multiplicity in the knot 0.25 is two, there is only C_1 continuity at that point.

A spline surface can be obtained as a tensor product spline,

$$F(u, v) = \sum_{i=1}^{n}\sum_{j=1}^{h} \beta^p_i(u)\,\beta^q_j(v)\,\mathbf{c}_{i,j}\,, \tag{1.4}$$

where $\mathbf{c} = \{\mathbf{c}_{i,j}\}$ is the set of control points. More information about splines can be found in the textbook [BGAA12].

1.3.2 Ruled surface

A ruled surface is a surface swept out by a line segment moving through space. In this thesis we use the end points to define the line segments. A ruled surface R is defined as

$$R(u, v) = (1 - v)s_2(u) + vs_1(u) \quad v \in [0, 1]\,, \tag{1.5}$$

where $s_1(v)$ and $s_2(v)$ are two parametrized curves following the end points of the line segment. In this thesis the curves s_1 and s_2 will be given as splines. For a degree 1 B-spline with no internal knots, we have $\beta_1^1(u) = (1 - v)$ and $\beta_2^1(u) = v$. Therefore equation 1.5 can be written as a tensor spline surface with degree 1 in one direction.

$$R(u, v) = \sum_{i=1}^{n} \sum_{j=1}^{2} \beta_i^p(u)\, \beta_j^1(v)\, \mathbf{c}_{i,j}\,, \tag{1.6}$$

where β_j^1 are B-splines of degree 1. Having multiple ruled surfaces combined at shared edges is known as a piece-wise ruled surface. This can be obtained by changing 2 to h in equation (1.6):

$$R(u, v) = \sum_{i=1}^{n} \sum_{j=1}^{h} \beta_i^p(u)\, \beta_j^1(v)\, \mathbf{c}_{i,j}\,, \tag{1.7}$$

where $h - 1$ is the number of surfaces that are combined.

A developable surface is a special kind of ruled surface where the tangent plane is constant along the ruling. More information about ruled surfaces can be found in the textbooks [Pre10, Zho10].

A hot wire always form a straight line, so the shape it cuts is a ruled surface. Cutting a piece-wise ruled surface with a hot wire, is obtained by cutting multiple times; the number of cuts needed is $h - 1$.

1.3.3 Curvature

The curvature is a measure for how much a curve bends or, more precisely, how fast the tangent rotates. For at planar curve the signed curvature can be calculated as:

$$\kappa(t) = \frac{\det\left(s'(t), s''(t)\right)}{\|s'(t)\|^3}, \tag{1.8}$$

where $s(t)$ is parametrized curve. If $\kappa(t) = 0$ for $t \in \mathbb{R}$ then $s(t)$ is a straight line.

When calculating the curvature for a given point on a surface, we look at the normal sections through the point. A normal section is a curve given by the intersection of the surface and a plane containing the surface normal at the given point. The curvature can be evaluated for each normal section using equation (1.8). The maximum and minimum curvature of these curves are the two principal curvatures, denoted by κ_0, κ_1.

The Gauss curvature is defined as $K = \kappa_0\kappa_1$ and reveals a lot about the surface. If $K = 0$ on every point on the surface, then it is a developable surface, which is a subset of ruled surfaces, and can therefore be cut by a hot wire. Examples of developable surfaces are a cone or a cylinder. If $K < 0$ then we are at a saddle point, and in special cases the surface is still ruled and can be cut. If $K > 0$ at any point on the surface, then a valley or hill are present, and therefore a hot wire

cannot cut the surface very well. More information on curvature can be found in the textbooks [Pre10, BGAA12].

Chapter 2

Cuttable ruled surface strips for milling

This paper proposes a novel preprocessing method for industrial robotic CNC-milling. The method targets a hybrid machining process, in which the main bulk of material is removed through robotic hot or abrasive wire cutting, after which regular CNC-machining is employed for removal of the remaining material volume. Hereby, the roughing process is significantly sped up, reducing overall machining time.

We compare our method to the convex hull, and remove between 5% and 75% more material, and on most models we obtain a 50% improvement. Our method ensures that no overcutting happens and that the result is cuttable by wire cutting.

2.1 Introduction

Recent years have seen a dramatic increase in the exploration of industrial robots for the purpose of architectural production [KGW14]. While predominantly still a topic of research, some of these developments have recently matured into commercialization, targeting the deployment of industrial robots for large scale production [Søn14]. Within subtractive processes, the Denmark start-up Odico has been successfully bringing robotic hot wire cutting to market.

CNC-milling is a well-established process in industrial production of, particularly, foam casting moulds, but also digitally produced stonework and bespoke timber manufacturing. While the process enables a very high degree of surface control and design freedom, it is also inherently limited by vast machining times for larger scale applications that require the removal of large quantities of material,

such as the machining of foam for ship hulls, wind turbine blades or architectural structures. This adversity becomes significantly amplified when applied to hard materials, such as CNC-milling of stone [SBB+16]. Wire cutting on the other hand, enables a dramatic reduction in production times, as volumetric artefacts can be produced in one or few swipes [MFS13]. However, here the precondition is that production geometries are described via ruled surfaces, which thus constrains the design freedom for the benefit of production efficiency. The wire cutting methodology and its implications are extendable to abrasive wire sawing of, for instance, stoneworks such as exemplified at the works of the Sagrada Familia cathedral [Bur16], as well as robotic abrasive wire sawing, as explored by [FS15].

The development of robotic hot blade cutting, [SFN+16], provides a cost-effective and time-efficient manufacturing process for general curved foam geometries. However, this process is also constrained by the detail level achievable, and is inadequate for small surface details, while well suited for large scale variations often deployed at industrial and architectural scale. In addition, so far, blade cutting is not applicable to non-foam materials.

Figure 2.1: Different cutting techniques: CNC-milling, hot wire cutting, and hot blade cutting.

The three processes milling, wire cutting, and blade cutting (se Figure 2.1) can be viewed as complementary, each covering a particular spectrum within subtractive machining. As such, an extension of the processes is to consider new ways for hybridization.

One such possibility is the combination of milling and wire cutting, in which the latter is applied for removal of that initial volume which would otherwise be machined in CNC-roughing processes. While CNC-roughing is generally fast compared to CNC-finishing, when assuming high target surface smoothness the roughing process can represent substantial machining times when applied to voluminous subtractions. Additionally, certain architectural applications may enable the omission of surface finishing machining in favour of leaving roughed surfaces for practical or aesthetic purposes. Two projects exemplifying this within the BladeRunner production portfolio, would be the [Fer14] (2014). In [Fer14], 212 m^3 of expanded polystyrene foam were milled to achieve a three-dimensional guideline shape to be coated in-situ with 70–100 mm of polished spent, giving the final shape, se Figure 2.2. Here, only roughing processes were applied as for ensuring enhanced binding between the foam core and concrete shell, roughing representing approximately 92

direct machining hours.

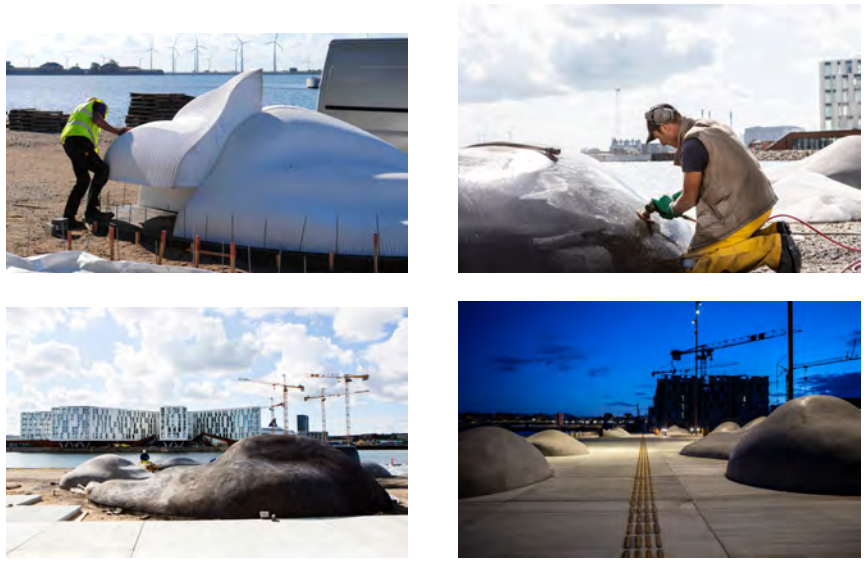


Figure 2.2: Creating an artificial concrete landscape in the urban harbor front of Copenhagen.

Given the amount of machining hours spend on roughing, a hybrid approach would, for this case, have caused a reduction in processing time of between 69–72%. In light of this finding, work was initiated to find a rationalization algorithm, which would cover any arbitrary three-dimensional shape with a set of non-convex ruled surfaces, such as to allow for a maximum of initial volume to be removed through wire cutting, while within the same robot cell shifting subsequently to a CNC tooling setup.

For this, we propose a method that combines fast wire cutting and precise CNC-milling removing as much material as possible using a wire before the precise shape is milled. As we allow multiple cuts the wire-cut surface is a piecewise ruled surface and it can be considered as an approximation or *rationalization* of the required surface. We can formulate the problem as follows: given a surface, rationalize it with a piecewise ruled surface such that the rationalization never intersects the original surface (no overcutting) and such that it can be manufactured by wire cutting. The latter implies that not only the rulings, but also the extension of the rulings, never intersect the surface.

Usually a mould is composed by several blocks and we do not consider the whole surface, but only a segment contained in a single block. As the final shape is milled we do not need to consider any continuity conditions between the piecewise ruled rationalizations of the different segments.

A ruled surface is given by moving a line segment through space while it changes length and orientation. As a line segment is determined by its endpoints the two

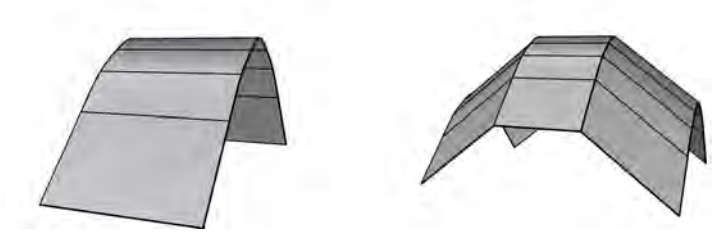


Figure 2.3: To the left a ruled surface defined by two curves. To the right a piecewise ruled surface defined by several curves.

curves described by the end points determine the surface uniquely, see Figure 2.3. A particular class of piecewise ruled surfaces is obtained by letting a polygon move through space while it changes shape, see Figure 2.3.

If the polygon at all times is on the outside of the original surface and furthermore is planar and convex then we are certain that the extensions of the rulings never intersect the original surface, see Figure 2.4.

Figure 2.4: Planar intersection of the surface and the piecewise ruled rationalization. If the rulings turn less than 180° then the convexity of the rulings guarantees that the extended rulings never intersect the surface.

Piecewise ruled surfaces are well known in architecture: [FNI+12] describes a method to rationalize free form architecture, focusing on the smoothness between rulings, and [FP10] find areas where a good rationalization can be done. Both papers use the asymptotic directions as guides for the rulings. In [WE14] large GPU powered dynamic programming is used to minimize the distance between the original surface and the rationalization.

The paper [EF97] constructs a piecewise ruled approximation of a free form surface using Bézier surfaces and a subdivision scheme to get the approximation within tolerated error, but global accessibility is not guaranteed. In [Elb95] a free form surface is approximated by piecewise developable surfaces, by using a simple developable primitive and a subdivision technique.

Milling with a cylindrical tool produces piecewise ruled surfaces, so they have also been studied in this context. To improve the tool path [Chi04] shows that

the error in the rough milling can be lowered by separating the ruled surface into multiple strips. The paper [CC06] constructs a piecewise ruled/developable rationalization where a subdivision scheme is used if a tolerance is not met. Tool interference is taken care of, but only to the extent of a fix axis flank milling tool. In [CD15] the one sided Hausdorff distance is used to minimize the overcutting.

The paper [JKS05] uses an iterative algorithm to automatically obtain rationalization consisting of developable patches. In [JTT+14] user input is used to create a Lobel mesh which has the utility to create developable patches.

Our method distinguishes itself by accepting a general free form surface as input and guaranteeing the wire does not cut into the model (overcutting) and that the rationalization is cuttable by a wire.

2.2 Method

Given a surface $\mathbf{f}$, we want to minimize the distance between it and a piecewise ruled spline surface $\mathbf{s}$ of the form:

$$\mathbf{s}(u,v) = \sum_{i=1}^{k}\sum_{j=1}^{h} \beta_i^p(u)\,\beta_j^1(v)\,\mathbf{c}_{i,j}\,, \tag{2.1}$$

where $\mathbf{c} = \{\mathbf{c}_{i,j}\}$ is the set of control points and β_i^p is a B-spline of degree p. Observe that $\mathbf{s}$ is a piecewise ruled surface since the basis function β_j^1 has degree 1.

We now discretize the piecewise ruled surface $\mathbf{s}$ by choosing a uniform grid (u_i, v_j), $i = 1,\dots,N$, $j = 1,\dots,M$, in the parameter plane and we discretize the original surface $\mathbf{f}$ by sampling points $\mathbf{f}_{i,j}$, $i = 1,\dots,N$, $j = 1,\dots,M$, on the surface. How the sampling is done is explained in Section 2.2.3 below. We furthermore make sure that the v-knots are among the parameter values v_j, i.e., we have indices $1 = j_1 < j_2 < \cdots < j_h = M$ such that the knot vector in the v direction is $v_{j_1},\dots,v_{j_h}$.

We measure the distance between $\mathbf{f}$ and $\mathbf{s}$ by the discrete square distance

$$\sum_{i=1}^{N}\sum_{j=1}^{M} \|\mathbf{f}_{i,j} - \mathbf{s}(u_i,v_j)\|^2, \tag{2.2}$$

2.2.1 Constraints

We need several constraints in the optimization, which we now describe one by one.

One sided approximation

To avoid overcutting the rulings should all be on the outside of the model. So if $\mathbf{N}_{i,j}$ is the *outward* normal of $\mathbf{f}$ at the point $\mathbf{f}_{i,j}$ then we require that

$$(\mathbf{s}(u_i,v_i) - \mathbf{f}_{i,j}) \cdot \mathbf{N}_{i,j} \geq 0\,, \quad \text{for all } i,j. \tag{2.3}$$

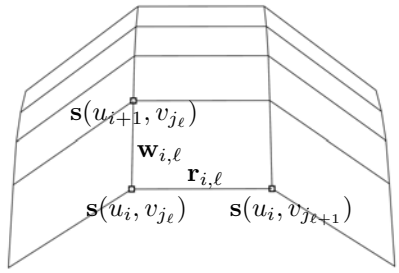


Figure 2.5: The discretized piecewise ruled surface from figure 2.3. The points $\mathbf{s}(u_i, v_{j_1}), \ldots, \mathbf{s}(u_i, v_{j_h})$ form an instance of the moving polygon. The vector $\mathbf{r}_{i,\ell}$ is a leg in the polygon, i.e., a ruling. The vectors $\mathbf{w}_{i,1}, \ldots, \mathbf{w}_{i,h}$ goes from one polygon of rulings to the next.

Planarity and convexity of rulings

The piecewise ruled surface $\mathbf{s}$ given by (2.1) can be considered as swept by a moving polygon and we require that the polygon $\mathbf{s}(u_i, v_{j_1}), \ldots, \mathbf{s}(u_i, v_{j_h})$ is planar for all i. We now let $\mathbf{r}_{i,\ell} = \mathbf{s}(u_i, v_{j_{\ell+1}}) - \mathbf{s}(u_i, v_{j_\ell})$, i.e., it is one of the rulings. The difference in the other direction is denoted $\mathbf{w}_{i,\ell} = \mathbf{s}(u_{i+1}, v_{j_\ell}) - \mathbf{s}(u_i, v_{j_\ell})$. The cross product $\mathbf{n}_{i,\ell} = \mathbf{r}_{i,\ell} \times \mathbf{r}_{i,\ell+1}$ is a normal to the plane spanned by $\mathbf{r}_{i,\ell}$ and $\mathbf{r}_{i,\ell+1}$, see Figure 2.5 . If all the normals $\mathbf{n}_{i,\ell}$, $\ell = 1, \ldots, h-1$ are parallel then the polygon is planar and if they all point in the same direction then the polygon is convex or concave. We can formulate this condition as

$$\mathbf{n}_{i,\ell_1} \cdot \mathbf{n}_{i,\ell_2} = \|\mathbf{n}_{i,\ell_1}\| \, \|\mathbf{n}_{i,\ell_2}\| \,, \quad \ell_1, \ell_2 = 1, \ldots, h-1 \,. \tag{2.4}$$

or to simplify it a bit

$$\mathbf{n}_{i,1} \cdot \mathbf{n}_{i,\ell} = \|\mathbf{n}_{i,1}\| \, \|\mathbf{n}_{i,\ell}\| \,, \quad \ell = 2, \ldots, h-1 \,. \tag{2.5}$$

To rule out the possibility of a concave polygon we require that the normal $\mathbf{n}_{i,\ell}$ points in roughly the same direction as $\mathbf{w}_{i,\ell}$. This can be formulated as

$$\mathbf{w}_{i,\ell} \cdot \mathbf{n}_{\ell,1} \geq \frac{1}{2} \|\mathbf{w}_{i,\ell}\| \, \|\mathbf{n}_{i,\ell}\| \,. \quad \ell = 1, \ldots, h-1 \,. \tag{2.6}$$

or as (2.5) secures that $\mathbf{n}_{i,\ell}$ points in the same direction as $\mathbf{n}_{i,1}$ we can simplify it to

$$\mathbf{w}_{i,1} \cdot \mathbf{n}_{i,1} \geq \frac{1}{2} \|\mathbf{w}_{i,1}\| \, \|\mathbf{n}_{i,1}\| \,. \tag{2.7}$$

Boundary

Ultimately the rationalized surface $\mathbf{s}$ is supposed to be cut from a block, which we assume has the form of a box with axis parallel sides, given by $a_1 \leq x \leq a_2$, $b_1 \leq y \leq b_2$, and $d_1 \leq z \leq d_2$. So no part of the boundary is allowed to be strictly

inside the block. We furthermore assume that we will be cutting roughly in the x direction. So we require that

$$c^x_{1,j} \le a_1\,, \qquad c^x_{k,j} \ge a_2\,, \qquad j = 1,\dots,h\,, \tag{2.8}$$

$$c^y_{i,1} \le b_1\,, \qquad c^y_{i,h} \ge b_2\,, \qquad i = 1,\dots,k\,. \tag{2.9}$$

where the superscript denotes the different components of the control points. In our implementation we start and end with the polygon on the block boundary, so in (2.8) the inequlities are replaced with equalities.

Limit the directions of the rulings

The rulings are not allowed to turn more than 180°, and this can be secured if the y-coordinate is a strictly increasing function. This is the case if it holds for the control polygon, and we formulate this as

$$c^y_{i+1,j} - c^y_{i,j} \ge \epsilon\,, \tag{2.10}$$

where ϵ is some small positive number. Strictly speaking we only need the difference to be non negative, but using an $\epsilon > 0$ also prevents any ruling from collapsing into a single point.

2.2.2 The optimization problem

We are now able to formulate the optimization problem

$$\begin{aligned}
&\underset{\mathbf{c}}{\text{minimize}} \sum_{i=1}^{N}\sum_{j=1}^{M} \|\mathbf{f}_{i,j} - \mathbf{s}(u_i,v_j)\|^2\,,\\
&\text{such that}\\
&c^x_{1,j} = a_1\,, \quad c^x_{k,j} = a_2\,, \quad j = 1,\dots,h\,,\\
&c^y_{i,1} \le b_1\,, \quad c^y_{i,h} \ge b_2\,, \quad i = 1,\dots,k\,,\\
&(\mathbf{s}(u_i,v_i) - \mathbf{f}_{i,j})\cdot\mathbf{N}_{i,j} \ge 0\,, \quad i = 1,\dots,k\,, \quad j = 1,\dots,h\,,\\
&\mathbf{n}_{i,1}\cdot\mathbf{n}_{i,\ell} = \|\mathbf{n}_{i,1}\|\,\|\mathbf{n}_{i,\ell}\|\,, \quad \ell = 2,\dots,h-1\,,\\
&\mathbf{w}_{i,1}\cdot\mathbf{n}_{i,1} \ge \frac{1}{2}\|\mathbf{w}_{i,1}\|\,\|\mathbf{n}_{i,1}\|\,,
\end{aligned} \tag{2.11}$$

where

$$\mathbf{n}_{i,\ell} = \mathbf{r}_{i,\ell}\times\mathbf{r}_{i,\ell+1}\,, \qquad \ell = 1,\dots,h-2\,, \tag{2.12}$$

$$\mathbf{r}_{i,\ell} = \mathbf{s}(u_i,v_{j_{\ell+1}}) - \mathbf{s}(u_i,v_{j_\ell})\,, \qquad \ell = 1,\dots,h-1\,, \tag{2.13}$$

$$\mathbf{w}_{i,\ell} = \mathbf{s}(u_{i+1},v_{j_\ell}) - \mathbf{s}(u_i,v_{j_\ell})\,, \qquad i = 1,\dots,k-1\,. \tag{2.14}$$

We solve this optimization problem using the interior point method, [WB05].

2.2.3 Initialization

All that is left is to explain how we choose the sampling points $\mathbf{f}_{i,j}$ and initialize the optimization.

First the coordinate system is chosen such that outward normal of the surface is roughly in the z-direction, i.e., upward. Then a cutting direction is chosen and we create N parallel planes orthogonal to the cutting direction and uniformly spaced. For each plane the intersection curve with the original surface $\mathbf{f}$ is found. Finally each intersection curve is discretized into M points. This is illustrated in Figure 2.6, where N and M both are 30.

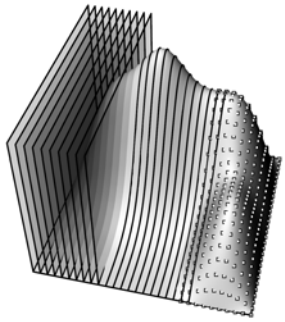


Figure 2.6: Illustration of the 3 steps that initialize the model into 900 points. Firstly: 10 of the 30 planes are shown, Secondly: 12 of the 30 intersection curves. Finally: 300 of the 900 discitization points are shown.

2.3 Results

We have run our algorithm on the five models shown in Figure 2.8.

Model 1 was cut in two different directions, four times in each directions, while Model 2 – 5 was cut four times in one direction. For comparison we have also calculated the convex hull, and the bounding box.

We have only considered cutting directions parallel to the sides of the blocks, but if we consider Model 2 diagonal cuts would be favourable. The depression in Model 4 poses a problem for wire cutting; no matter what direction a line at the depression has, it will cut into one of the 'mountains' at the edge. On the other hand we see that even though we sweep the surface using a polygon with four legs the optimization has put two of the legs on the same line. So we have in effect a polygon with only three legs and consequently only need three cuts to produce the rationalization.

If we imagine the surface sitting inside the block and remove the outer part we are left with a solid object, see Figure 2.7.

No.	Model	Ruled	Convex Hull		Bounding Box	
1	0.4924	0.0374	0.1541	76%	0.5076	93%
2	0.2435	0.1520	0.3380	55%	0.7565	80%
3	0.5914	0.0201	0.0520	61%	0.4086	95%
4	0.3058	0.1379	0.1453	5%	0.6942	80%
5	0.4533	0.0798	0.2294	65%	0.5467	85%

Table 2.1: In the second column the volume of the model is shown. In column 3,4, and 6 we show how much volume there is left for milling using our method, the convex hull, and the bounding box respectively. All volumes are normalized with respect to the volume of the bounding box. In column 5 and 7 we show how much more volume our method removes compared to the convex hull and the bounding box, respectively.

We have calculated the volume of that object in each case. The results are summarised in Table 2.1.

We *normalize* all volumes with respect to the volume of the bounding box, i.e., we use Vol / Vol(BB). We show the volume of the model, but in the three other cases we show the volume that needs to be milled away, i.e., $(\text{Vol} - \text{Vol(Model)})/\text{Vol(BB)}$.

So for Model 1 we can see that the volume of the model is 49% of the bounding box volume, that our method has left 4% of the bounding box volume for milling, that the convex hull leaves 15% of the bounding box volume for milling, and that the bounding box (doing nothing) leave 51% of the volume for milling. We also show how much more volume our method removes compared to the convex hull and the bounding box, respectively. That is we show $(\text{Vol} - \text{Vol(our)})/(\text{Vol} - \text{Vol(Model)})$. For Model 1 this is 76% and 93% respectively.

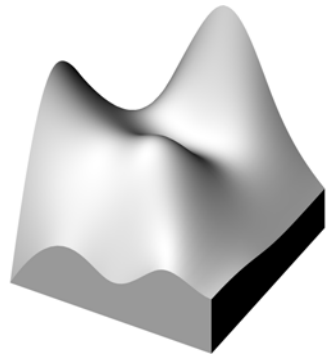


Figure 2.7: Model 1 with volume shown, the volume is created by intersection the bounding box with the surface

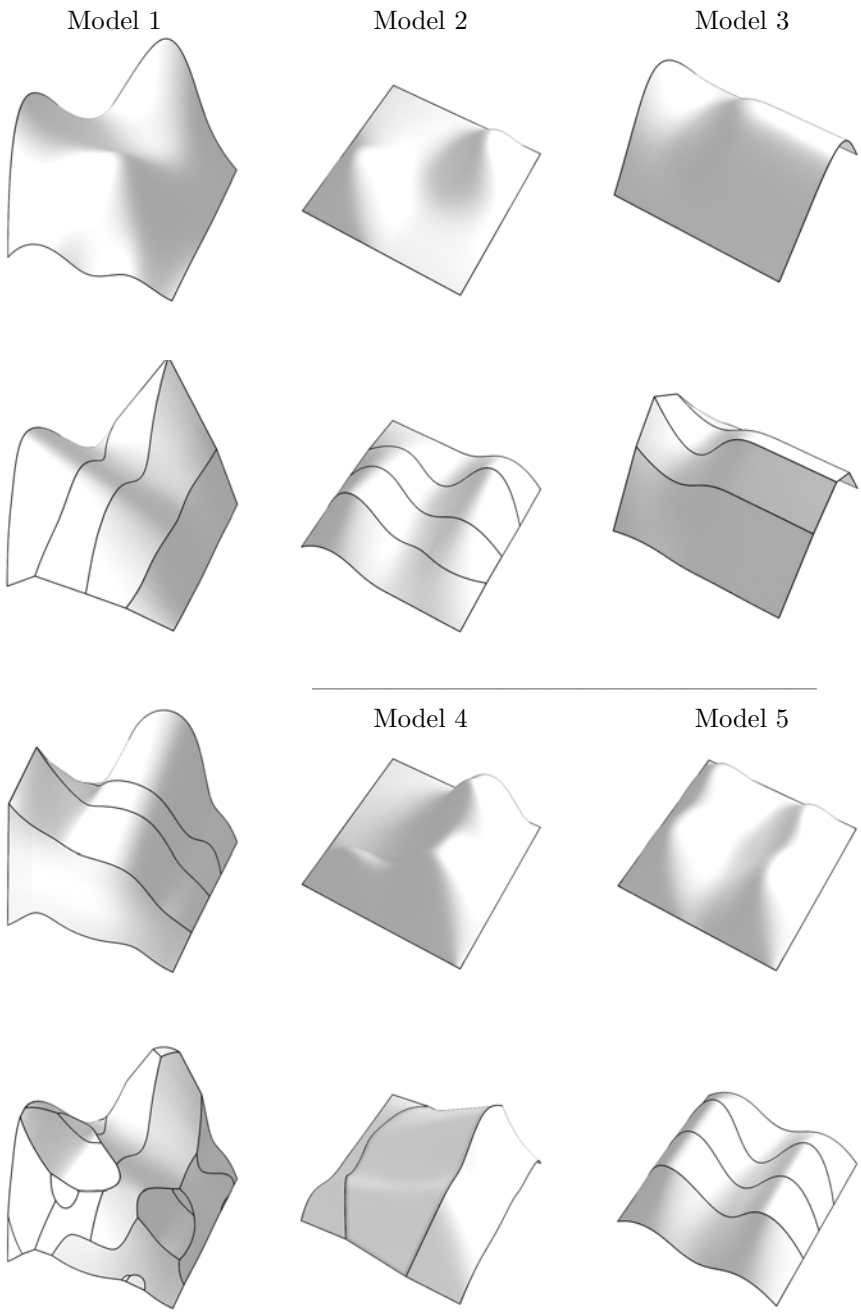


Figure 2.8: Five models rationalized by piece-wise ruled surfaces. Model one has been cut four times in one directions and four times in roughly the orthogonal direction. The four other models have been cut four times in one direction.

2.4 Conclusion and future work

We have described a novel method that finds a one sided approximation of a free form surface by a piecewise ruled surface. The method guarantees that no extension of the rulings cut through the original surface. This allows us to use the method and wire cutting as a preprocessing step for milling.

Compared to using the convex hull as a preprocessing step we obtain an improvent from 5 to 75% and typically around 50%.

For simplicity we have limited ourself to cuts parallel to the coordinate planes, but relaxing this conditions and allowing any cutting direction will improve the result.

We represent the piece-wise ruled surface as a tensor product spline surface of degree one in one direction and the planarity condition of the rulings restrict the flexibility of the piecewise ruled surface. To overcome this we can choose another representation where we explicitely move a planar polygon through space.

In this work we have assumed that the full architectural model has been segmented into block sized portions, and we have only considered the piece inside a single block. An interesting possibility is to use our method to aid in the segmentation. If we allow for non-convex polygons in the optimization we will obtain a better fit and we could use the concave vertices to guide the segmentation.

Chapter 3

Block Segmentation

This chapter is a manuscript titled *Block segmentation* and written together with Andreas Bærentzen, Jens Gravesen, and Alla Sheffer. This work is one of the main contributions of my PhD.

3.1 Introduction

While the technologies collectively known as *3D printing* have not disrupted mass manufacturing as such, the impact on many industries has been profound. Using *3D printing* for prototypes of small physical objects are now easy to make, architectural models can be produced directly from CAD designs. Bespoke objects like hearing aids are simply printed, and objects – whose complex shapes preclude that they could ever have been physically removed from an injection mold – are quite easily manufactured with *additive processes*.

Despite these great strides forward, 3D printing of large scale objects remains a challenge. In the realm of *robotic architecture*, there have been several projects aimed at building through robotic brick stacking [DSG+11], timber construction[1], or fused deposition modeling using concrete.

These efforts are important, but they do not necessarily point toward a general family of methods that would be useful for 3D printing any object of, say, a cubic meter or larger; this remains very difficult and expensive.

Within architecture, the use of *subtractive manufacturing* is probably the best approach. For instance, large formwork is made possible by casting concrete in molds created through machining expanded polystyrene (EPS). More recently, cut-

[1] http://www.spezialschalungen.com/

ting EPS with a hot blade has opened up the possibility of realizing fairly large physical objects with positive curvature – based on digital models – both fairly quickly and inexpensively. However, these processes are still limited by the size of an EPS block (often about $50 \times 50 \times 50$ cm). Thus, actual designs are often split into several blocks. This is straight forward as long as it is reasonable to use a regular arrangement of blocks as would be the case if the design is for a wall or some other highly regular structure. In the general case, a regular arrangement of blocks is often an exceedingly bad choice that would lead to far too many blocks, many of which barely intersect the surface.

Fundamentally, all subtractive and additive manufacturing processes have an inherent limit to the size of the object to be produced due to the physical limits of the equipment. We propose a method that divides a 3D object into regions which are feasible for production. Furthermore, we lower the amount of material used, by minimizing the number of building blocks need for construction. We have been motivated by manufacturing using hot wire cutting of expanded polystyrene. Combined with our method, this process allows the manufacturing of extremely large objects. However, our method is generic and could also be used with other techniques for manufacturing.

3.2 Previous work

Approximation of a 3D volumetric model is a sparsely researched subject. The authors of [YIO+15] uses sticks to create buildings, and the authors of [SFLF15] segment a model into interlocking parts that are easy to 3D print. The authors of [SDW+16] create an internal polyhedral base of a model and combine it with an external shell of 3D printing. Even though these articles present solutions to volumetric problems, none of the methods can be used for large scale fabrication in the industry.

Approximating 3D surfaces is a more developed field where large scale structures are build using different methods. The authors of [ZLAK14] approximate surfaces with Zometool building bricks that have strict limitations on how they can connect. The authors of [FNI+12, TSG+15, FP10] approximate surfaces by either glass tiling or ruled surfaces for fabrication.

We use stochastic optimization in this paper, which is related to work in [ZLAK14] and [MVLS14] that uses stochastic optimization to solve surface approximation and layering problems, respectively. However, they do not consider gaps in the layers or consider the geometry under the surface.

Hexahedral meshing is used to improve fabrications; see [LSVT15] for state of the art hexahedral meshing. hexahedral meshing algorithms need to have good looking hexahedral inside the models as well as on the surface, since they are used for large FEM simulations, and therefore need to consider not only the surface. Even though quad meshing terminology covers some of the same areas as this paper, the quads are small and can vary in size, and therefore the methods do not apply to the same building constraints.

Packing problems have similarities to the problems dealt with in this paper, but the concepts of allowing overlap and prohibiting gaps is not within the frame of the packing problem. The packing problem is a NP hard problem that has been extensively studied in the literature. The article [BLM10] describes a method for packing identical rectangles into a larger rectangle without overlapping. The authors of [Pis05] use dynamic programming to solve the knapsack problem in pseudo-polynomial time. The article [MPV00] describes the 3D case of the knapsack problem.

3.3 Method

Let the model M be a closed mesh in $\mathbb{R}^3$ contained in a collection of blocks $\mathbf{b}_i \subset \mathbb{R}^3$, $i = 1, ..., N$. The goal is to remove as many blocks as possible while keeping the surface contained in the (rearranged) collection of the remaining blocks. That is, we have the following optimization problem

$$\begin{aligned} &\underset{\substack{\text{Placement of } \mathbf{b}_i, \\ \text{i=1,...,N}}}{\text{minimize}} && N\,, \\ &\text{subject to} && M \subseteq \bigcup_{i=1}^{N} \mathbf{b}_i\,. \end{aligned} \tag{3.1}$$

We allow each block to be translated and rotated in $\mathbb{R}^3$, giving six degrees of freedom per block. Hence we have a high dimensional, non-linear, discrete optimization problem. This calls for stochastic optimization methods.

We initialize the optimization by calculating the minimum-volume bounding box, [BHP99], of the model and aligning a 3-dimensional grid of blocks to it. As a first step we remove all blocks that do not intersect the model, see Figure 3.1. For

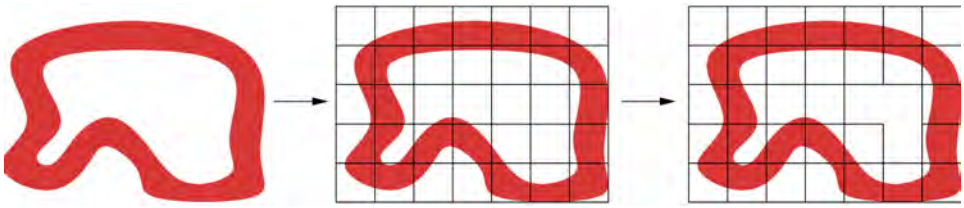


Figure 3.1: 2D example of the initialization. Left: the model, middle: a grid of blocks are laid over the model, and right: non-intersecting blocks are removed.

simplicity we use equal size cubes for the blocks, but the algorithm can easily be generalized to general blocks. We can also allow blocks of different sizes, but then the initialization has to be changed.

3.3.1 The idea

The basic idea behind our algorithm is that if the intersection between a given block and the model is completely covered by the other blocks, then the chosen block is superfluous and can be removed.

The objective function, N, in the optimization problem (3.1) is discrete and does not detect whether a small change is benificial or not. We therefore add a continuous term that promotes larger overlaps between blocks and hence increses the chance of removing a block. This leads to the smooth objective function:

$$E = N - E_1 = N - \frac{\sum_{i=0}^{N} \sum_{j=i+1}^{N} 2\Omega(\mathbf{b}_i, \mathbf{b}_j)}{N(N-1)V + \epsilon}, \tag{3.2}$$

where $\Omega(\mathbf{b}_i, \mathbf{b}_j) = \text{Vol}(\mathbf{b}_i \cap \mathbf{b}_j)$ is the volume of the overlap between two blocks, and V is the volume of a single block. Observe that we have $0 \leq E_1 < 1$, so removing a block is always better than creating more overlap. The ϵ in the denominator ensures that $E_1 < 1$, otherwise we would have $E_1 = 1$ when all blocks are exactly on top of each other.

We want to have some block completely covered by other blocks. So one large overlap is better than many small overlaps. We can promote that by introducing a $p \geq 1$, and consider the new objective function:

$$E = N - \frac{\sum_{i=0}^{n} \sum_{j=i+1}^{n} (2\Omega(\mathbf{b}_i, \mathbf{b}_j))^p}{N((N-1)V + \epsilon)^p}. \tag{3.3}$$

For the examples in this paper we have used $p = 2$ and $\epsilon = 0.00001$.

3.3.2 Pure translation

Let us consider a model in $\mathbb{R}^3$, see a 2D example in Figure 3.2 left. We wish to translate the blocks to increase the overlap between the blocks, to be able to remove a block entirely, as described in Section 3.3.1.

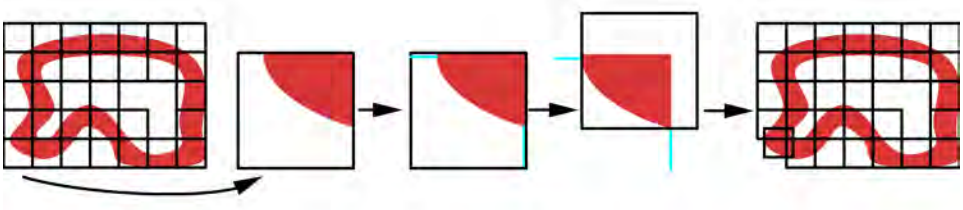


Figure 3.2: How translation works on a 2D example.

We also need to satisfy the constraints at all time. So we pick a block $\mathbf{b}_i$, find the part of the model that is not covered by other blocks, $M_i = \mathbf{b}_i \cap (M \setminus \bigcup_{j \neq i} \mathbf{b}_j)$. For each direction, x, y, and z, parallel to the edges of the block, we can translate a maximal amount, x^+ and x^- say, without violating the constraint, $M_i \subseteq \mathbf{b}_i$, in

the positive and negative direction, respectively. We then perform the translation given by $(x^+ - x^-)\mathbf{e}_x + (y^+ - y^-)\mathbf{e}_y + (z^+ - z^-)\mathbf{e}_z$, where $\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$ are the unit vectors parallel to the edges of the block, see Figure 3.2.

3.3.3 Rotation

To cover features that are aligned in a non-coordinate direction it is advantageous to rotate the blocks. So suppose we are given a block, $\mathbf{b}_i$. Again we consider the part of the model, M_i, that is not covered by other blocks. We then calculate a minimal bounding box of M_i and check its dimensions. If one of the sides has a length that is larger than the side length of the block, the block cannot cover the bounding box and we do nothing. Otherwise the bounding box is smaller than the block and we can rotate the block such that it is aligned with the bounding box, see Figure 3.3. We then translate the block such that the bounding box is in one

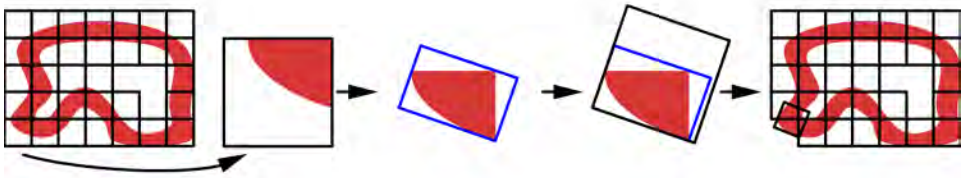


Figure 3.3: How rotation works on a 2D example.

of the corners, see Figure 3.3. In 2D there are four posibilities and in 3D eight.

3.3.4 Alignment

This operation is similar to the rotation operation, but here we choose not one, but a collection of m neighbouring boxes, $\mathbf{b}_{i_1}, \ldots, \mathbf{b}_{i_m}$, see Figure 3.4. The collection,

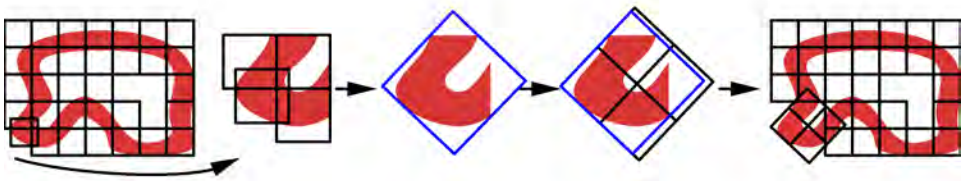


Figure 3.4: How alignment works on a 2D example.

$I = \{i_1, \ldots, i_m\}$, is chosen as a particular block and all blocks that intersects it. We then determine the part of the model that is not covered by other blocks, $M_I = \bigcup_{i \in I} \mathbf{b}_i \cap (M \setminus \bigcup_{j \notin I} \mathbf{b}_j)$. As before, we calculate the minimal bounding box of M_I and calculate how many blocks that are needed to cover it, m' say. If $m' > m$ we do nothing, otherwise we remove $m - m'$ blocks and arrange the remaining blocks

$\mathbf{b}_{i_j}$, $j = 1, \ldots, m'$ in a regular grid aligned with the bounding box and translated so they cover the bounding box, see Figure 3.4.

3.3.5 The algorithm

Lets consider the movement operations above: pure translation and rotation both act on a single block and create a more compact segmentation (i.e. with more overlap), where alignment acts on multiple blocks and create a more dispersed segmentation (i.e. with less overlap). Therefore we use pure translation and rotation until the objective function is in a local minimum, and then align the segmentation to reset the search space in hopes of finding a better local minimum. Blocks are never added, so even though we leave a local minimum, the solution never gets worse.

For each block, taken in random order, we consider the one translation operation and the eight rotation operations. Of the nine operations, we perform the one that lowers the objective function most. If none of the operations improve the objective function, no action is performed. If a block is moved, we check if any of the other blocks affected by this can be removed. This procedure is repeated until a cycles of the blocks do not improve the objective function. In this case we assume the segmentation is in, or close to, a local minimum, and may proceed to alignment. The alignment procedure traverses the blocks once in random order. If a block has been part of an alignment operation, it is removed from the list of blocks to traverse.

The alignment procedure, followed by the rotation/translation procedure, is repeated until two subsequent alignments do not remove any blocks; in this case the algorithm terminates.

3.4 Results

The algorithm has been applied to several different models, spanning from a simple torus to an octopus with eight arms. The results of the segmentations for eight models are shown in Figure 3.5.

The segmentation of the algorithm, at different iterations, are shown in Figure 3.6: the initial state, two intermediate states, and the result. An iteration refers to a cycle through all the blocks, either in the translation/rotation procedure or in the alignment procedure. In this example the algorithm will test around 1500 movement operations in each iteration; however, this scales with the number of blocks in the segmentation. The number of blocks removed is highest in the first 20 iterations and then decline. This is because it becomes harder too remove a block, the fewer blocks there are. The algorithm terminates after 77 iterations, but in iterations 49 to 77, no blocks are removed, and thus any of these segmentations can be used as the result. This example is a typical case with regard to how the number of blocks removed decreases with iterations.

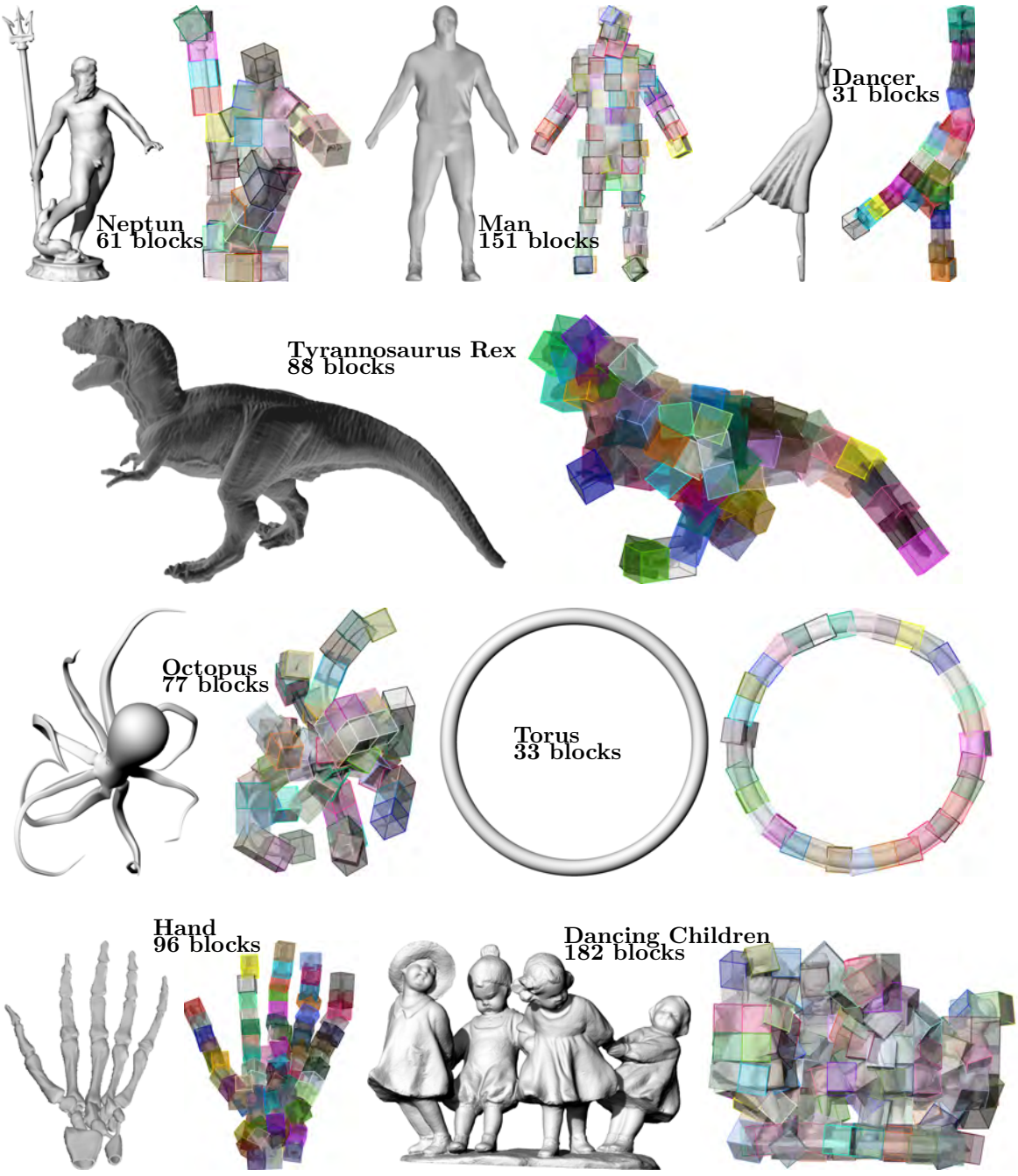


Figure 3.5: Eight models and the segmentation of them.

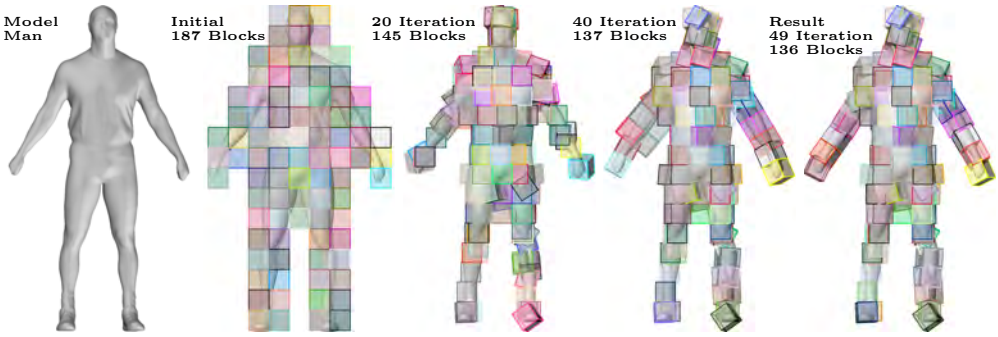


Figure 3.6: The model and four stages of the segmentation. From left to right: the model; the initialization; after 20 iterations; after 40 iterations; the result after 49 iterations. In one iteration, each block is been considered at least once.

Model	No B Init	No B Result	Percentage
Torus	52	33	37 %
Octopus	115	68	41 %
Neptun	107	76	29 %
Neptun2-3	143	103	28 %
Neptun3-2	87	61	30 %
Fertility	222	200	10 %
Dancing children	204	182	10 %
Dancer	60	31	48 %
Man	187	136	28 %
Hand	147	96	35 %
Tyra	118	88	25 %
Armadoli	148	117	20 %

Table 3.1: The number of blocks used in the initial state and in the final segmentation for twelve cases. The reduction in percentage is also shown.

A comparison between the number of blocks in the initial state and in the result is seen in table 3.1. The algorithm removes between 10% and 37% of the blocks, which is a significant reduction. The largest improvement is for models with long slim structures that fit into the dimensions of a block, such as the *Torus* and *Octopus* models. The improvement is smaller in models with large compact regions with a box-like shape, such as the *Fertility* and *Dancing Children* models.

The *Neptun* model is segmented for three different block sizes, to illustrate how this affects the algorithm. Defining the blocks in the *Neptun* model to have a volume of 1, the blocks in the *Neptun2-3* model have the volume $\frac{2}{3}$, and the blocks in the *Neptun3-2* model have the volume $\frac{3}{2}$. The percentage of blocks removed is nearly the same for the different block sizes, see Table 3.1. In Figure 3.7 is a comparison of the three segmentations; the *Neptun3-2* segmentation looks more clean even though it is only marginal better than the others. In all three cases the

block segmentation follows the underlying geometry of the model.

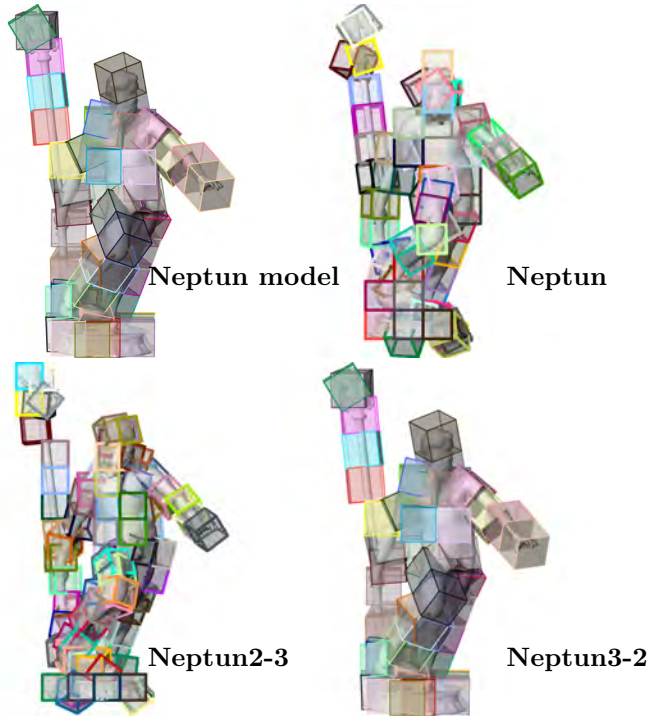


Figure 3.7: The *Neptun* model and its segmentation for three different block sizes.

The examples with the *Neptun* model shows the effects of changing the block size on relatively small scale. But how do the problem change if the block size is changed on a larger scale? If the block size is increased to the point where only about ten blocks are needed, then the blocks could just as well be placed manually. If the blocks size is scaled down, then the segmentation would eventually become a surface problem and not a volumetric problem. Therefore it might be more effective to use surface algorithms such as [TSG+15] when the segmentation reaches around 1000-10000 blocks, depending on the geometry, even though it is not a problem for our algorithm to handle such a segmentation.

3.5 Design choices

In this section we discuss some of the choices in the algorithm. In particular, we analyze the three operations in more detail.

A segmentation of a model aimed for fabrication is only usable if it covers the entire model and does not leave gaps. Our method ensures the segmentation fulfills this constraint, and thus is a viable solution, at all times. When developing the algorithm, we attempted other strategies that allowed gaps during the optimization.

These methods either resulted in a lot of small gaps that needed to be covered after the algorithm terminated, or produced an abundance of blocks during the algorithm trying to cover the gaps that appeared. In both cases the resulting segmentations had more blocks than the initial state, and thus these methods were not usable.

The three operations have different effects on the segmentation, and to investigate these effects, we utilize that the operations can be applied independently. We compare four different combinations of the operations: translation only, rotation only, translation and rotation, and the full algorithm (i.e., translation, rotation and alignment). The results are shown for one model in Figure 3.8 and for four models in Table 3.2.

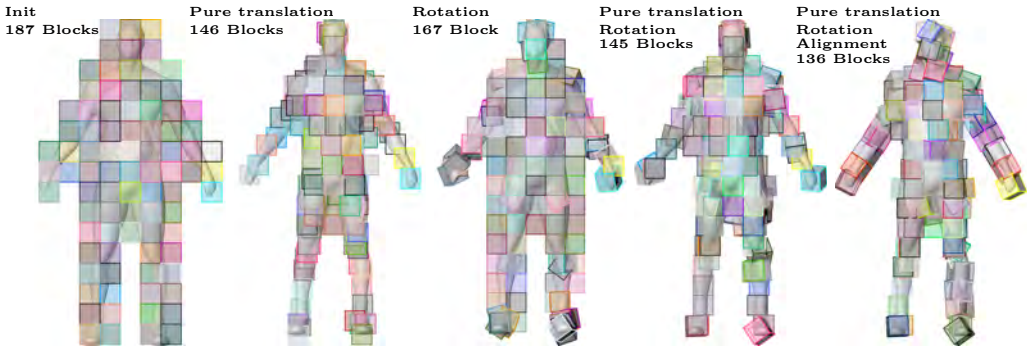


Figure 3.8: The initial state and the result of different combinations of operations. The rightmost case corresponds to the full algorithm.

From Table 3.2 we see that pure translation works quite well alone, whereas rotation alone does not lower the number of blocks very much. Both operations serve the purpose of increasing the overlap, by pushing blocks into the model, but when all the blocks are aligned the pure translation works much better than the rotation. If a block is translated towards another aligned block, it will produce an overlap proportional to the length of the movement. Initially, when all the blocks are aligned and there is no overlap, the minimal bounding box determined for rotation movement is often too large for the block to cover. As a consequence, we observed in Figure 3.8, that most of the blocks are not moved from the initial state when rotation is applied alone. The rotation operation is more suited when the segmentation is more chaotic, where it can align the blocks to the geometry of the model. In two of the examples pure translation alone actually produces a better result than pure translation and rotation combined, suggesting we need to be mindful of how to include rotation.

Adding the alignment procedure (going from pure translation and rotation combined to the full algorithm) improves the result for all four models, see Table 3.2. In the optimization there are many local minima with respect to the translation and rotation operations, and the alignment allows the segmentation to settle in another minimum.

For all models the full algorithm gives the best result, which indicates that all

Model	Initial state	Translation	Rotation	Translation and rotation	Full algorithm
Man	187	146	167	145	136
Octopus	115	73	88	78	68
Armadilo	148	121	137	123	117
Neptun	107	94	94	76	76

Table 3.2: The number of blocks in the initial step, and for different combinations of the operations.

three operations have their merits. Pure translation appears to be very effective in reducing the number blocks, and is also the simplest of the three operations. Alignment has the effect of dispersing the segmentation and creating order in the chaos again. Rotation can align the segmentation to the geometry, especially in chaotic segmentations. For a specific model it might be beneficial to try different combinations of the operations in varying order.

3.6 Conclusion

We have described a novel approach to perform volumetric segmentation of free form architecture. The focus was on lowering the material needed to construct the models, and our method achieved up to 48% reduction of blocks in the segmentation.

Chapter 4

Segmentation versus rationalization

To construct a 3D model, we need a segmentation of the model into blocks, and a recipe for cutting each block with a hot wire or hot blade. In this chapter we consider an algorithm for finding this for a given model. We focus on hot wire cutting, which implies the surface of a block needs to be approximated with a piece-wise ruled surface, though this does not ensure the block is cuttable. We consider the case where milling and other post-processing tools are not used, so the aim is for the result to be as close to model as possible, and also pleasing to the eye. To make the final result smooth, $\mathcal{C}^0$ continuity between blocks is required. We begin by discussing two approaches to rationalization and segmentation, mentioned in [BBaE+].

4.1 Challenges

If the CAD model is ruled, all the blocks will be ruled regardless of their position, and there is no need for rationalization. On the other hand, if the model is not ruled, we can use two different strategies for rationalizing it. Either each block is approximated separately or the entire model is approximated with a piece-wise ruled surface. With the first approach the model is segmented into blocks and then each block is rationalized. With the second approach the entire model is first rationalized; this segments the model into patches, where each patch is a ruled surfaced. Then the rationalized model is segmented into blocks. The two strategies are illustrated in Figure 4.1, where a half sphere model is segmented into two blocks and rationalized. In the middle column each block is approximated with a ruled surface, following the first approach. The second approach is shown in the right

column, where the entire half sphere is approximated with a single ruled surface (corresponding to a single patch) and then divided into two blocks.

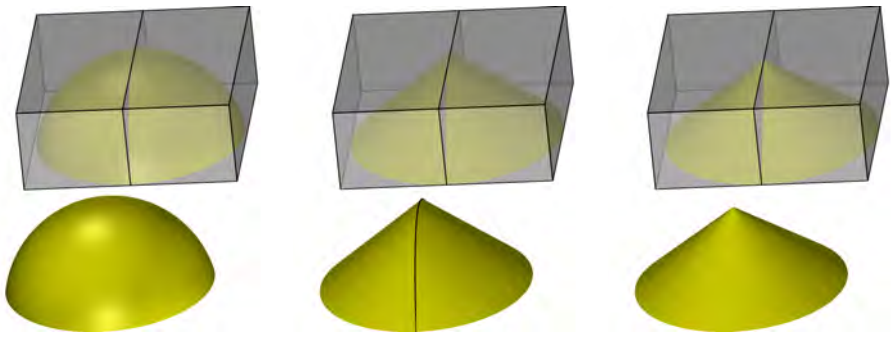


Figure 4.1: A half sphere model segmented into two blocks and rationalized in two ways. The top row shows the blocks. Left column: The model. Middle column: Each block approximated separately. Right column: The entire model approximated with a single ruled surface.

It turns out that approximating each block separately can result in some undesirable artefacts. This is not so obvious and is easiest to illustrate with an example, so consider the model in Figure 4.2. The model is segmented into a grid of blocks and we start to rationalize each block, see middle image. However, a problem emerges, when we attempt to rationalize the 16th block, which in the figure has no rulings. There is an artefact on the right side of the block, where the rulings will have a different direction from the neighboring block. In some cases the problem can be avoided by choosing the direction of the rulings carefully, but in other cases this is not possible. We can make the artefact disappear by requiring the rulings on the neighboring block to lie on a straight line and rationalize this block again. Unfortunately, this gives a rigid system where it is necessary to rationalize a block several times. Another option is to use global information when rationalizing a block. On the other hand, such artefacts do not occur if the entire model is approximated with a piece-wise ruled surface, see Figure 4.2 right, where the model is approximated with 14 ruled surfaces. Afterwards the rationalized model can be segmented into blocks, and regardless of how the blocks are placed, each block will contain a piece-wise ruled surface. Therefore, the appearance of the final result does not depend on the block segmentation, except for segmentation seams.

Deciding which approach to use is often a question of what is considered optimal in the final result. There is a huge difference between optimizing for the minimal number of cuts and aiming for a result that is pleasing to look. We present a method where the goal is to have a smooth model, and the strategy is to rationalize the entire model and then segment it into blocks.

Figure 4.2: Illustration of the artefacts that can occur when rationalizing each block instead of the entire model. Left: A half sphere model. Middle: The model have been segmented into blocks and we have started to rationalize each block. The rulings are shown as black. The block with no rulings have neighbors with conflicting directions of the rulings. Right: An approximation of the entire model with a piece-wise ruled surface containing 14 patches.

4.2 Algorithm

Before the model is rationalized, it is important to have good starting guess, and to get a good approximation, the surface needs to be close to a piece-wise ruled surface. We achieve this by using a mesh algorithm that divides the model into regions. These regions are approximated by piece-wise ruled surfaces, which are combined to create a rationalization of the entire model. The rationalized model is segmented into blocks using the algorithm from Chapter 3, resulting in a set of blocks with piece-wise ruled surfaces. However, this does not necessarily imply that the shape of the blocks can be cut with a hot wire, and thus a post-processing step is added.

To make a good initial setup for the rationalization, we create a Polar–Annular Mesh (PAM) of the model, from [BAS14]. The PAM is a quadrilateral mesh and is created from a standard mesh. See Figure 4.3 for an example on the Armadillo model. The PAM divides the mesh into different regions, represented with different colors. The dark blue dots are the dividing line between the regions and the red dots are the unregular points. Each region have a structure that resembles a cylinder or a cone. For the Armadillo model the horns, fingers, tail, snout, and toes resemble cones, and the hands, neck, legs, and arms resemble cylinders. Both cylinders and cones are ruled primitives. This makes each region in the PAM suited for approximation into piece-wise ruled surfaces.

The approximation of each region in the PAM is computed by minimizing the squared distance between a region, $\mathbf{f}$, and a piece-wise ruled surface, $\mathbf{s}$, for a set of discrete points:

$$\min \sum_{i=1}^{N} \sum_{j=1}^{M} \|\mathbf{f}_{i,j} - \mathbf{s}(u_i, v_j)\|^2 . \tag{4.1}$$

This corresponds to the method in section 2.2 without the constraints. The rationalization of the PAM into a piece-wise ruled surface is shown for the Armadillo

Figure 4.3: Left: The Armadillo model. Middle: A Polar–Annular Mesh of the Armadillo model. Each color represents a different region, except the dark blue that are region borders and the red dots that are unregular points. Right: Piece-wise ruled surfaces rationalization of each region.

model in Figure 4.3.

Next we segment the rationalized model into block using the algorithm from Chapter 3. Each block now has a piece-wise ruled surface, but unfortunately the blocks might not be cuttable; for more details of what makes a surface cuttable, see [SNS+]. The process of making the blocks cuttable have not been carried out for the Armadillo model, but is illustrated using a cross-section of an Armadillo arm as a 2D example, see Figure 4.4. Notice that some of the blocks cannot be cut, since an extension of a ruling (representing the wire) would penetrate the model. If a block is not cuttable, taking the convex hull of the model area covered by the block, gives a cuttable shape. This is illustrated to the bottom left in the figure, where the red lines indicates the added material from the convex hull.

The block segmentation method creates solution where the blocks will often overlap, and this overlap may be used to improve result of making the surface cuttable. The overlap could be distributed between the blocks, such that the least amount of material is added, when taking the convex hull for each block. An example is shown in Figure 4.4 bottom right, where the overlap between the green and yellow block is divided at the purple line. As this example shows, dividing the overlap with a line through the intersection of two rulings is optimal, if the two rulings together create a concave shape. An interesting outlook is to utilize the optimal way to distribute overlap, to create a block segmentation that reduced the material added for the blocks to be cuttable.

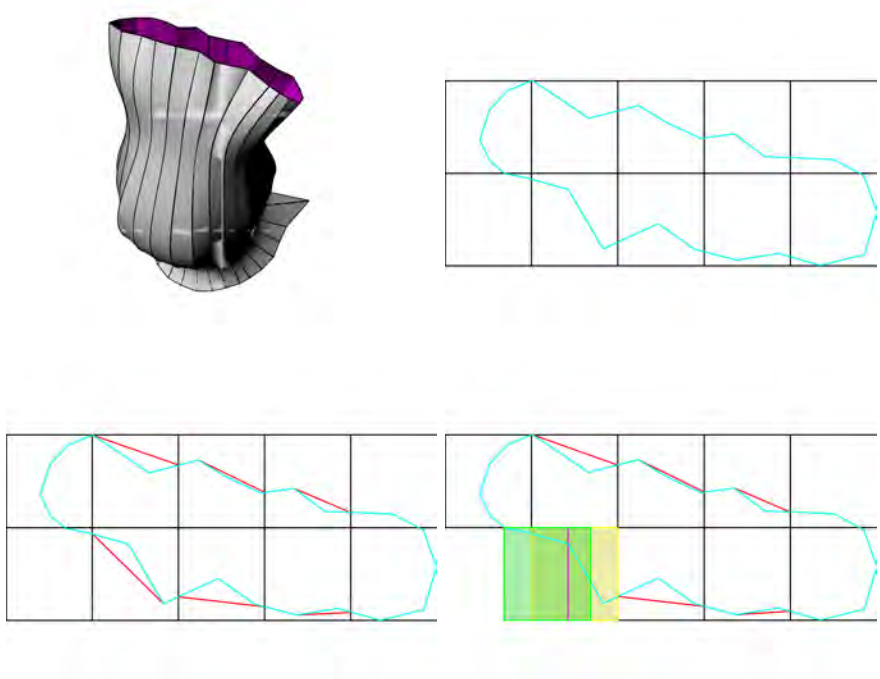


Figure 4.4: Going from a piece-wise ruled surface to a cuttable shape. Top left: Armadillo arm approximated with piece-wise ruled surfaces. Top right: Cross-section of the arm with blocks. Bottom left: The red lines show where the convex hull for each block add material. Bottom right: The green block has been moved as part of the block segmentation method, creating overlap with the yellow block. By dividing the overlap at the red line, both blocks become cuttable without adding material.

Chapter 5

Elastica segmentation

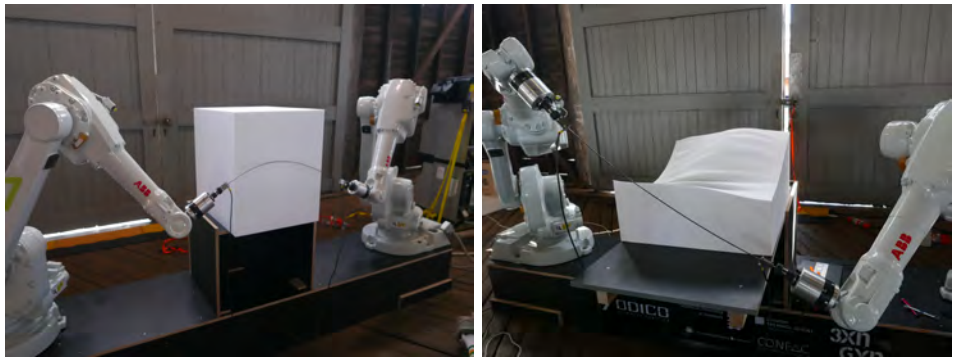


Figure 5.1: Two different hot blade shapes, produced by the same robot. Picture are from a three days workshop at ROBARCH 2016 organized by the BladeRunner team.[1]

One of the goals of the BladeRunner project is to explore the possibility of using a hot blade (or a hot tube) cutter instead of a hot wire cutter. In contrast to the hot wire, which is taut all the time, the hot blade can bend. The shape of the hot blade is determined by the robot controlling the position and angle of the blade at the end points. An example of the robot holding the blade curving upwards is seen in Figure 5.1 left. The hot blade is moved through a block of EPS. The robot can change the position and the angle at the end points, while melting through the block, giving large variations in the models that can be produced. In Figure 5.1 right the block has been cut once and is ready for a second cut. Multiple cuts

[1] http://www.robarch2016.org/workshops/ Superform: Robotic hot blade cutting workshop

extend the possible variations of the models. The hot blade's ability to positively curve increases the possible shapes compared to a hot wire.

All the possible shapes of the blade belong to family of curves known as Euler elastica. An Euler elastica is defined as the curve that minimizes the energy of an elastic rod in a plane, which is equivalent to minimizing the squared curvature over the curve,

$$\int_{\zeta} \kappa^2(s) ds, \tag{5.1}$$

where κ is the curvature, and ζ is the curve domain. The mathematics of Euler elastica is described in tha PhD thesis chapter 5 [Lev09]. Often a desired curve for the blade is given by a spline and we need to approximate it with an elastica; an analytic method for this can be found in [BGN15] and a numerical method in [BBC+]. I will use these methods throughout this chapter. An example of an Euler elastica is shown in Figure 5.2 in blue. When the blade's shape is approximated only a subsection of the infinitely long elastica is used; this is represented by the red part of the curve. Another approximation might use another subsection of the same elastica or another elastica all together.

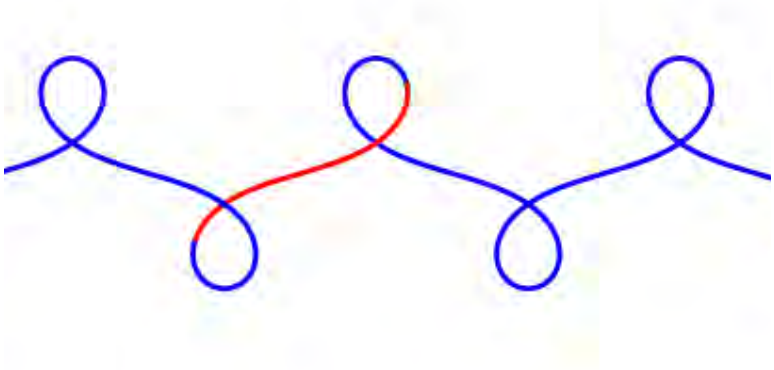


Figure 5.2: An Euler elastica curve in blue. The red subsection of the curve is a possible shape of the blade.[2]

In this chapter I present different algorithms for solving the problem of segmenting a model, such that it can be cut by the robotic setup introduced above. To be able to cut the segmentation five production constraints need to be met; in section 5.1 these constraints are described and how they relate to the physical setup. Due to the case studies designed in the BladeRunner project, the algorithms described in this chapter are conducted on facades, implying one side of the models are facing a building wall. This gives a limited set of cutting directions for the robots, and a clear indication of what is up and down on the models. We have

[2]Picture created by Toke Bjerge Nørbjerg

developed three segmentation algorithms: The inflection points algorithm, in section 5.2, is a method where the model segmentation is governed by the number and position of inflection points. The trace algorithm, in section 5.3, focuses on the aesthetic aspect of segmentation by considering the valleys and ridges of the model geometry. The longest elastica algorithm, in section 5.4, tries to find the largest region of the surface that can be approximated by elastica, and uses this information for segmentation.

5.1 Robotic setup and constraints

We want a hot blade robot to cut a block into a given model of a facade, the hot blade is restricted to always keep the hot blade in a plane.

In most cases, we discretise the model into N curves by taking the intersection between N parallel planes and the model, see Figure 5.3. Each curve is approximated by an elastica, representing the shape of the blade. Between the given elastica curves, the robot uses an interpolation method that we do not control. Therefore the shape of two neighbouring elastica curves should be relatively close. If each of the N planar curves are well approximated by elastica, we assume the result is a good approximation of the surface. The core concept of all the algorithms is making N parallel planar curves and approximating them with elastica, but some of these restrictions will be relaxed in more advanced algorithms, see section 5.1.5.

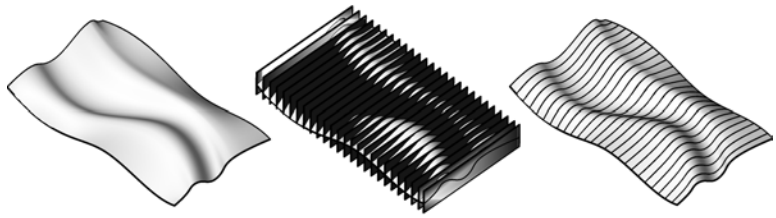


Figure 5.3: Left: The original facade. Middle: 26 parallel planes evenly distributed. Right: the 26 planar curves of the facade ready to be approximated.

If the facade is simple enough, the hot blade can cut the model from a single block. However, if the model has high curvature, the construction is large, or the intersection curve is not approximated well by an elastica, it is necessary to segment the model. We will describe five production constraints that governs when segmentation is need.

5.1.1 Dimension constraints

The physical dimensions of the model, the building block, and the hot blade imposes some natural constraints. If the model is larger than the building block, or the approximated elastica curve is longer than the blade, segmentation is required. These are hard constraints and can only be circumvented by acquiring a longer

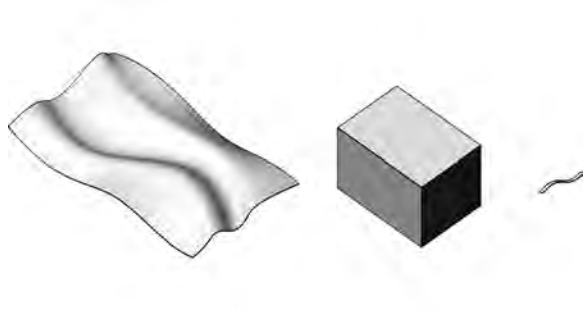


Figure 5.4: Left: model surface. Middle: building block. Right: hot blade.

blade or a larger block to cut from. Observe in Figure 5.4 where the model is clearly larger than the building block, which, in turn, is larger than the blade; here segmentation is needed, both of the model and the building block.

5.1.2 Approximation constraint

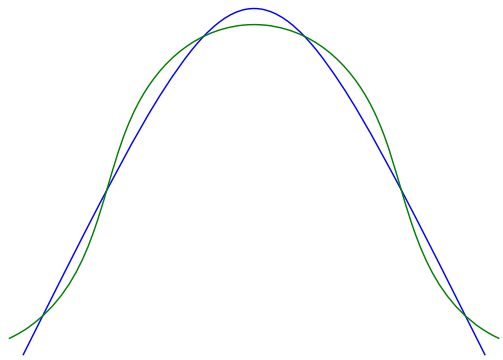


Figure 5.5: Blue line is the desired curve (a spline). Green curve is the approximated elastica. [3]

This constraint focus on how well the elastica approximates the intersection curve. As a measure of the approximations accuracy, we use the squared L2 distance between the curves [BGN15]. To cut the model accurately, it is necessary to find an elastica curve that approximates the desired curve well, since the blade can only attain elastica shapes. If this is not possible it is necessary to segment the model.

[3] Picture taken from article [BGN15]

5.1.3 Shape constraints

These constraints concern how easily the blade can keep the intended shape, and if the movement of the blade leads to an ambiguous shape.

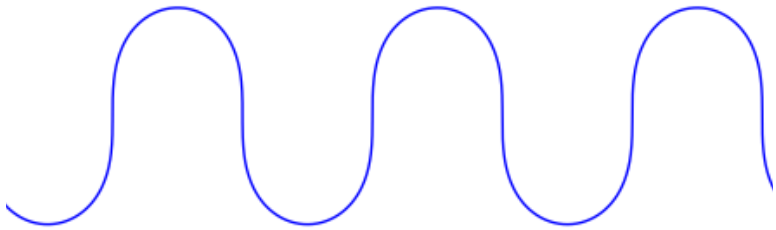


Figure 5.6: Illustration of an elastica shape that the blade cannot keep. [4]

The difficulty of attaining and keeping a shape of the blade increases with the number of inflection points, which are points where the curvature changes sign. For example, bending a plastic blade as in figure 5.6 is far from easy. Even if the blade has such a shape, a small movement can easily cause it to unravel into a shape with fewer inflection points suddenly and uncontrollably. Therefore we impose a maximum of two inflection points for the elastica curve; otherwise segmentation is needed.

Since an elastica curve has continuous curvature, an intersection curve has the same number of inflection points as its elastica approximation. Two exceptions to this are inflection points near the end of the curve, which may disappear or appear, and pairwise inflection points close to each other, which may cancel, just like pushing out a small dent. Segmenting the model by placing a partitioning through the inflection point, does not remove the inflection point.

We find the inflection point by calculating the signed curvature $k(t)$ for M points on each curve $c(t)$, $t = 0, ..., M$:

$$k(t) = \frac{\det\left(c(t)', c(t)''\right)}{\|c(t)'\|} \tag{5.2}$$

If two neighbouring points have curvature with opposite sign, there is an inflection point between them. Such points are denoted by $p_{t,dt}$, where t is the lower t value of the pair, and dt is the parameter distance between them. To find a more precise estimate of the inflection point, bisection is employed: we iterate

$$p_{t,dt} = \begin{cases} p_{t,dt/2} & \text{if } \operatorname{sgn}(k(t+dt/2)) = \operatorname{sgn}(k(t+dt)) \\ p_{t+dt/2,dt/2} & \text{otherwise} \end{cases} \tag{5.3}$$

until dt is under a desired tolerance.

[4] Picture created by Toke Bjerge Nørbjerg

Another issue arises if the blade is straight and its endpoints are moved towards each other; then it is impossible to predict whether the blade will bend upwards or downwards. This can be fixed by minutely changing the tangent at one end point. Segmentation, however, will not help solve this problem, so it is not considered in the algorithms.

5.1.4 Blade strain constraint

If the blade is bend beyond a certain amount, it can be permanently deformed. This will occur if the curvature of the model is to high in intersection curves. An example of this can be seen in figure 5.7, which was takend at a workshop presenting hot blade cutting. The blade is heavily deformed because the model designs exceeded the maximum curvature. Unfortunately this cannot be helped by segmenting, since the new segments will have approximately the same curvature as the original segment.

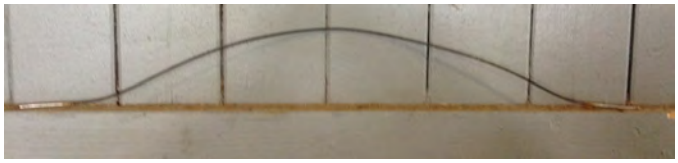


Figure 5.7: An 80 cm long hot tube, that was heavily deformed after one day of cutting. The tube was used to cut models with curvature radii down to 9 cm, far below what it could withstand. Observe how the ends are still straight, because the robot held it there.[5]

5.1.5 Multiple cuts constraints

When cutting with a blade in one direction and keeping the blade shape in parallel planes, the part of the blade which is cutting can change. The front end and the back end of the blade have the same shape, but are shifted horizontally. When the blade moves upwards while cutting, the front of the blade cuts the block, while the backside cuts when the blade moves downwards.

This issue can be resolved by using a tube instead of a blade, having the algorithm compensate for the width of the blade, or rotating the intersection planes. The latter option is shown in Figure 5.8, and is an alteration of the general setup, where all the planes are parallel. A cutting direction is given as a curve c on the surface of the model; observe how the planes follows this curve, marked with red in the Figure 5.8. The normal of each cutting planes is equal to a tangent of c,

$$c'(i\,dt) = p_i^n, \tag{5.4}$$

where $i = 0, ..., N$, dt is the curve distance between each plane, c is a parameterised curve between 0 and $N\,dt$, and p_i^n is the normal of plane i.

[5] http://www.robarch2016.org/workshops/ Superform: Robotic hot blade cutting workshop

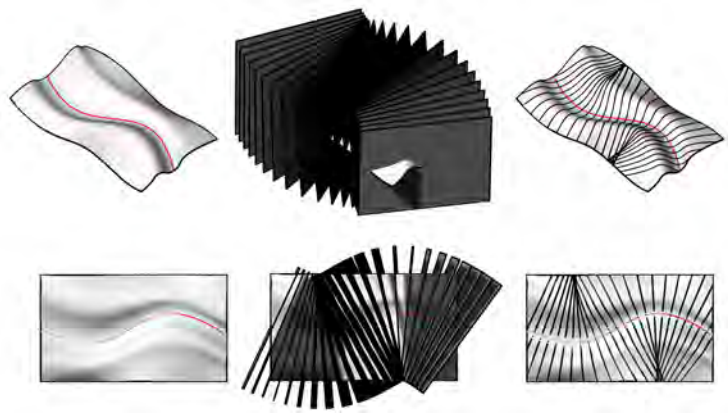


Figure 5.8: Left column: the model where the red curve c is the cutting direction. Middle column: the rotated planes found by equation 5.4. Right column: the curves are the intersection between the planes and the model. The two rows show different view points.

Rotation is a further prospect in the robot setup, and therefore only relevant for more advanced cutting in the future. Unfortunately it entails that the blade cuts the same plane multiple times, which is undesirable. This can be seen in Figure 5.8(right), where the intersection curves overlap. In such cases, segmentation is needed. In some of the segmentation algorithms described below, rotating the planes is addressed as a side note.

5.1.6 Limitation of the constraints

It would be optimal to incorporate all these constraints in a single algorithm. Unfortunately, to test if the approximation constraint 5.1.2 is fulfilled, we need to approximate a large part of the model surface, and this is slow. Therefore, we separate the segmentation algorithms into two categories:

a) The slow and reliable algorithms, that repeatedly approximate the surface until all the constraints are met. These algorithms always give a result that can be cut.

b) The fast and unreliable algorithms, that handles most constraints and hope to automatically fulfilled the rest. If some constraints are not fulfilled after such methods, simple post-processing heuristics are applied until all constraints are met.

One way of looking at it, is that category **a** algorithms use the elastica approximation to guide the segmentation, whereas category **b** algorithms use the segmentation to guide the elastica approximation.

5.2 Inflection points algorithm

This algorithm belongs to category **b** and incorporates the shape and dimension constraints. It is my contribution to the paper [SFN+16]. The problem solved in this algorithm is: given a facade model, how to segment it into blocks, such that each block fulfills the mentioned constraints.

To initialize the segmentation a grid of blocks, with M rows in the cutting direction and N columns in the other direction, are placed such that they cover the entire model surface. We assume all the blocks have a fixed size. To give the algorithm flexibility, blocks in the same row are allowed to overlap. An overlap is not physical possible, and in reality it represent that excess material are cut away from one of the blocks. The algorithm attempts to fulfill the shape constraint by adding blocks or moving blocks within a row. If the blocks are small enough to be cut by a blade, the dimension constraint is automatically fulfilled.

The general setup described in section 5.1 is applied, where the model is discretized with N intersection curves lying in parallel planes. To fulfill the shape constraint, the curves can have maximum two inflection points in each block; or more precisely, the part of each block that is left after overlap is removed. The inflection points are computed for all the curves, and the curves with more than two inflection points are marked, as shown in Figure 5.9.

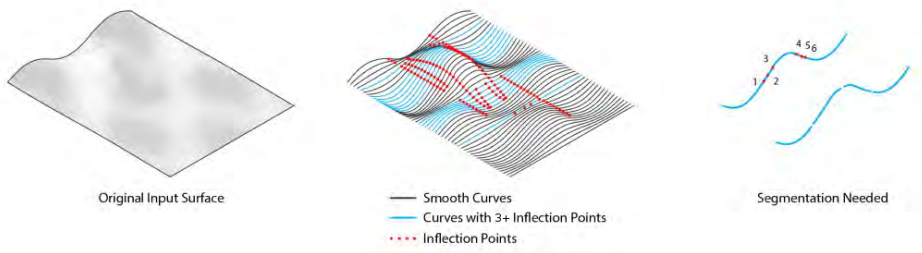


Figure 5.9: Left: Model surface. Middle: Hot blade planar cuts with inflection points. Right: Close up of two cuts. Points 5 and 6 could potentially cancel each other, depending on the scale.[6]

When the inflection points are computed, each of them are assigned to the block that covers them. If a points is covered by two blocks, it is only assigned to the block that still covers it after overlap is removed. If a block b has assigned more than two inflection points on a curve, then one of two options are performed:

1. A copy of block b, denoted b_c, is inserted. The copy is translated in the row such that the area $b \backslash b_c$ is as large as possible, while only containing curves with maximum two inflection point. An illustration of this can be seen in the two left images of Figure 5.10.

[6] Picture from [SFN+16], with correction.

2. Block b is translated along the row so it decreases its overlap with another block. It is translated as far as possible without the other block is assigned to many inflection points. This is to distribute the inflection point better between them. See the two right images in Figure 5.10.

Option b is always tried before a, to minimize the number of block added to the segmentation. All translation are perpendicular to the cutting direction. This procedure is repeated until no block has a curve with more than two inflection points.

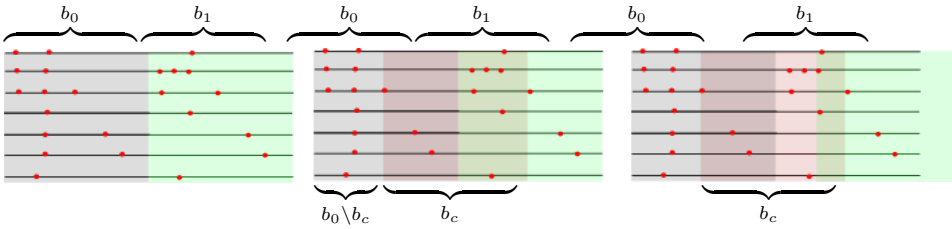


Figure 5.10: The algorithm seen from the above with inflection points in red and intersection curves in black. Left: The initial state of b_0 in gray and b_1 in green. Middle: Step (a) for block b_0. The copy b_c of b_0 is added and moved to the right until $b_0 \backslash b_c$ would have a curve with three inflection points. Right: Step (b) for block b_1. Block b_1 is translated to the right until $b_c \backslash b_1$ would have a curve with three inflection points.

An example of running the algorithm is seen in Figure 5.11. To the left is the model surface and to the right the resulting segmentation from the algorithm. In this example the segmentation is symmetric, implying all rows are identical. This was enforced by translating movement and addition of blocks in one row to all the other rows as well. Furthermore, the segmentation curves were adjusted to follow the geometry of the surface in the overlap between the blocks.

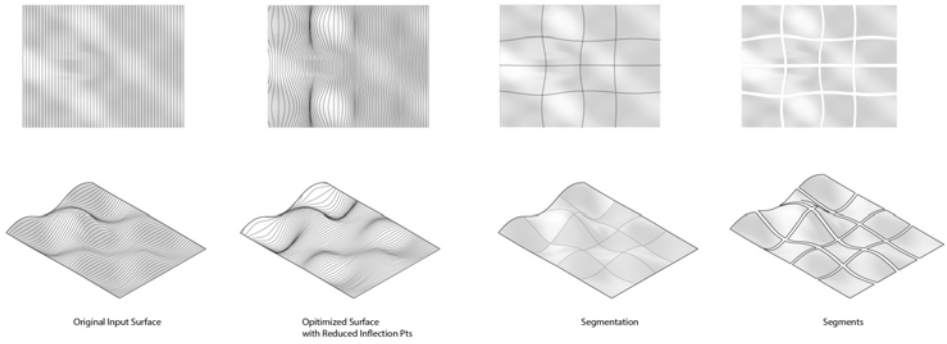


Figure 5.11: From left to right: First, the inititial state of the algorithm. Second, the initial state with rotation of the planes. Third and fourth, the final result of the segmentation.[7]

[7] Picture from article [SFN+16]

As a side note we consider the idea of adding rotation of the planes (section 5.1.5), which can be seen in figure 5.11 second from the left. In such an algorithm additional checks for curve overlap need to be done when a move option is performed.

5.3 Trace algorithm

This algorithm belongs to category **b** and handle the dimension and shape constraints. The problem solved in this algorithm is: given a facade model, how to segment it in into regions that follows the model geometry, such that each region fulfills the constraints. The algorithm described in this section is a collaborate work between Toke Bjerge Nørbjerg and me. The segmentation idea sprang from a joint venture in the BladeRunner project on producing the model shown in figure 5.12. The model was designed by 3XN as the first test case for hot blade cutting in the BladeRunner project. Currently, the model is in production, and the plan is to put it on display as an example of hot blade cutting.

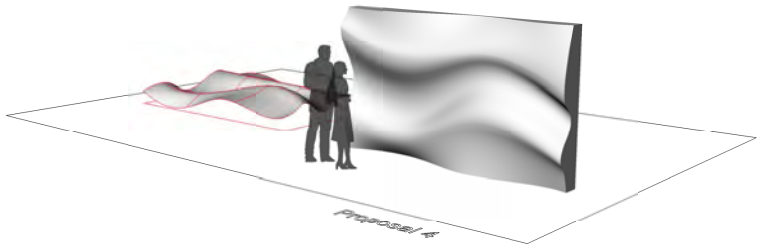


Figure 5.12: The test case model designed by 3XN. Observe the scale in comparison to the silhouettes.[8]

The concept of the algorithm is to let the segmentation follow natural valleys and ridges of the model geometry along the cutting direction. Since the segmentation is visible in the final product, this helps to ensure a natural look of the model. We are not concerned with segmentation in the cutting direction, since it is only governed by the block dimension constraint, and thus it can be applied independently of segmentation in the other direction. The basic idea of the algorithm is to use local maxima of the curvature (of the intersection curves) to represent the valley and ridges, even though this is not the local maxima or minima of the curves. At an inflection point the curvature attains the minimum value zero, so between two inflection point the curvature must have (at least) one local maximum (unless it is a straight line). By letting the segmentation follow curvature maxima, the inflection point constraint is automatically fulfilled.

The general setup described in section 5.1 is applied, where the model is discretized with N intersection curves lying in parallel planes. The inflection points

[8]Model created by 3XN http://www.3xn.com/

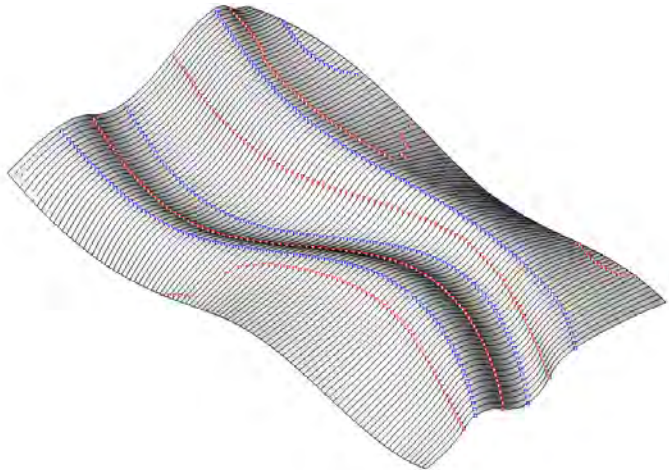


Figure 5.13: The blue points are the local maxima of curvature for each curve, the red points are the inflection points for each curves. Both the red and blue points form curves on the surface.

are computed for all the curves, and the curvature maxima between them are found, see Figure 5.13. There are six pairs of inflection point with no curvature maxima between them; presumably this is because of numerical inaccuracy. The curvature maxima are connected to form curves on the surface; we will refer to these curves as traces. The traces can have end points on the edge of the surface, in the middle of the surface, or they can form closed curves. The traces for the test case are shown in Figure 5.14 left, and their end points all lie on the edge of the model surface.

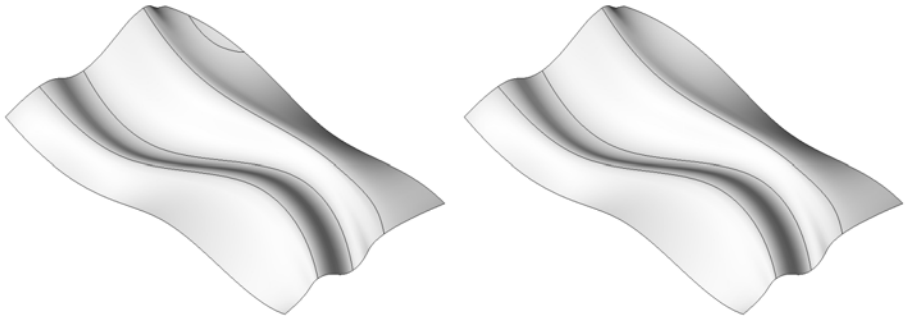


Figure 5.14: Left: The traces of curvature maxima. Right: The result after a short trace is removed. The traces follow the geometry of the model.

The traces on the model constitutes the segmentation, and therefore we do not want a trace with end points inside the model surface. Each trace ending inside the model is extended, starting with the longest trace, until it meets the end of another trace, which then becomes a part of the original trace, or it reaches the edge of the model. A trace is extended such that the ratio, between the distance to its two

neighbor curves, is kept constant. Here neighbor curves are the two closest curves, taking from the set of already extended traces and the two edges in the cutting direction. If two traces come so close that inflection points between them can be smoothed out, the traces are merged. For the test case no extension was needed since all the traces ended on the edge of the model.

If the model have many short traces before the extension procedure is performed, the algorithm will produce a segmentation with many small segments. Therefore a model with many short traces is smoothed, and the traces on the smooth model is projected onto the original model. If inflection point disappears in the smoothing process, we assume they can be smoothed out in the final curve.

On the other hand there can also be too few traces to fulfill the dimension constraint. This implies the distance between traces are larger than the block width or larger than the length of the blade. In this case additional traces are added, dividing the band between the original traces down the middle, see Figure 5.15 with an example for the test case.

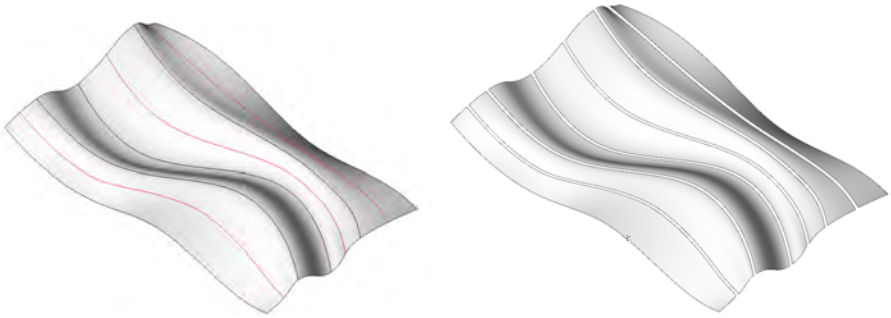


Figure 5.15: Left: The original traces in black, and extra traces in red. Right: The 7 segments of the result.

For the test case the algorithm works great and creates smooth curves that follow the model geometry very well. This will also be the case for similar models with long smooth features. This algorithm is not designed to handle traces that forms closed curves, and therefore we cannot guarantee it will work for arbitrary forms of geometry.

This is an algorithm that aims for artistic segmentation, which I personally think is an interesting field with great potential. This type of algorithms also have merits in the architectural environment, where computer aided design software are commonplace. Thus segmentation algorithms can aid the architect design process.

5.4 Longest elastica algorithm

This algorithm is in category **a**, where elastica approximation guides the segmentation, and it ensures that the dimension and approximation constraints are fulfilled. The algorithm described in this section is a collaborate work between Toke Bjerge

Nørbjerg and me. As with the trace algorithm in section 5.3 we only consider the segmentation across the cutting direction. The concept of the algorithm is to start at a model edge that goes along the cutting direction, and find the largest region that can be approximated well by elastica curves, i.e. fulfills the approximation constraint, see section 5.4. After a region is found, it is removed from the model and the algorithm starts again from the new edge of the now smaller model.

The general setup, described in section 5.1, is applied, where the model is discretized with N intersection curves lying in parallel planes. For each intersection curve c we determine the longest subsection of c, which have a good elastica approximation. The idea is to iteratively divide the curve until an accepted solution is found, see Figure 5.16. The subsection starts at c_0, which is fixed to the starting point of c, and ends at c_1. Initially, c_1 is chosen so the length of the subsection is the maximum cutting length of the blade, denoted L. The initial subsection is approximated by an elastica, and if the approximation meets the demands, the subsection of c is accepted; otherwise we iterate over $i = [1 \dots M]$. For each step i, if the elastica is acceptable, c_1 is moved $\frac{L}{2i}$ away from c_0 along the curve. On the other hand, if the elastica is not acceptable, c_1 is moved $\frac{L}{2i}$ towards c_0 along the curve. After M iterations, the last c_1, which resulted in an accepted elastica, represents the solution of finding the longest subsection of c that starts at c_0.

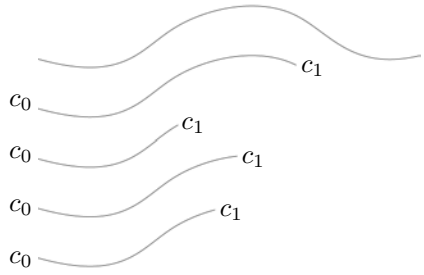


Figure 5.16: Three iteration for finding the longest accepted elastica curve. From top to bottom: the full length curve c; the first subsection with length L (not accepted); the second subsection (accepted); the third subsection (not accepted); the fourth subsection. Observe the extension or subtraction of the curve is halved for each step.

This subdividing method assumes that if a curve is approximated well, then all subsections of the curve are also approximated well. In each iteration the step length of c_1 is halved, and thus c_1 never attains the same value twice. The number of iterations controls the precision: For example, if L =1m, a precision of 1cm requires $M = 7$ iterations, and a precision of 1mm requires $M = 10$ iterations. If all approximations fail, the final curve between c_0 and c_1 will have a length under the desired precision.

When the longest elastica subsection of each intersection curve c is computed, all the c_1 points are connected to create a segmentation of the model. Unfortunately, the subsection length can vary a lot, which is not desirable, since the final segmentation line should be a smooth. Therefore, a post-processing step is added,

to ensure that no subsection curve are more than a maximum distance longer than its neighbors. The subsections curves are considered in order from shortest to longest, and if the neighbors of a subsection curve are too long, the neighbors are cut. A final segmentation line is now obtained by connecting the end points of the shortened subsections.

To construct the next segmentation, the whole procedure is repeated with the starting point, c_0, of the curves, moved to the edge of the segmentation that was just found. This is repeated until there are no more model to segment.

The result of applying the algorithm to the same test case as in section 5.3 can be seen in Figure 5.17, although we have the changed the cutting direction. This was to lower the maximum curvature of the blade, because the blade strain constraint was more restrictive than originally assumed. In this example, the algorithm starts at the left edge of the model, and the segmentation curves become increasingly curved as the algorithm moves toward the right. This is because a few curves repeatedly have short subsections, but their neighbors are allowed to be longer, and this effect is cumulative.

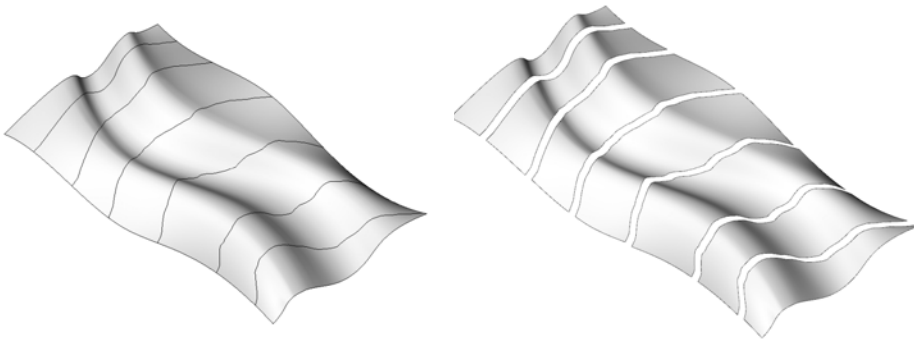


Figure 5.17: Left: The result of the algorithm starting from the top left edge. Right: The segments of the result.

In contrast with the inflection points algorithm (section 5.2) and the trace algorithm (section 5.3), the longest elastica algorithm in this section ensures that the cutting curves can be approximated by elastica. However, the many computations of elastica approximations makes the algorithm costly in terms of run-time. The algorithm does not consider the aesthetics of the segmentation in regard to the model geometry. To make the segmentation more pleasing to the eye, the end points of the subsections could be connected by a smooth curve rather than straight lines.

The algorithm could be developed further to also account for the shape constraint, by also restricting the length of the subsection curves to include a maximum of two inflection points. This extended algorithm could be used to make a general scheme for finding a segmentation that incorporates all constraints. First it would be necessary to determine a cutting direction, where the blade strain constraint was fulfilled, i.e. the blade is kept within a maximum curvature. The multiple cut

constraint could be fulfilled by using a hot tube and not allowing for rotation of the intersecting planes. This would guarantee a segmentation that was ready for fabrication by robots.

Chapter 6

Discussion

The stochastic optimization developed in chapter 3 allow us to segment buildings into the minimum number of blocks, thus saving up to 48% of the material needed for construction. This lead us to expect that the material saving algorithm will be of great value in the industry as it is still a relatively untouched area in the scientific field. Since this is an automatic segmentation, the seams are not controlled by the architect and this can alter the overall design of the structure. We believe that small alterations of the algorithm could force the segmentation into a specific pattern, thus guiding the design into a more aesthetic result. This algorithm can also be used as the base for other methods where aesthetic considerations are taken into account, since the algorithm always produces a feasible design.

Rationalization of piece-wise ruled surface to lower the cost of milling was considered in chapter 2 and the result was a method reducing the material needed to be milled by up to 95%. The method was highly constrained, and a more relaxed method could give a more flexible result. A relaxed version of the algorithm could also be used as a segmentation method if only a few ruling pairs were relaxed. Many of the constraints was applied on the ruled surface and not on the control grid. Shifting the constraints to the control grid will make the system more flexible. From a design perspective, the method is probably best suited for cutting in hard material like marble, since ruffing in soft material like EPS is quite fast.

When cutting form works using elastica there are many constraints that need to be taking into account and this was addressed in chapter 5. The architectural designs currently being produced have sufficiently low curvature to automatically ensure that many of these constraints are uphold. This means the elastica segmentation algorithms are trying to solve the obstacles in advance. Since hot-blade cutting is still in the development phase, obstacles not yet encountered will arise in

the future, but the method can be a stepping stone for further development. The segmentation algorithm also aims for a aesthetic result, though it is only a small subset of the tools that architects can choose from when designing.

Many of the methods developed have the purpose of lowering the cost of production. One of the project's goals was to give architects a tool that would allow them to pick their design from a cost perspective; meaning that an architect should have the possibility to see the end result of their design for different price categories (with different production methods) during the design process. Like picking the correct tiles for a floor: both considering price and style.

Chapter 7

Conclusion

I have presented six algorithms for rationalization and segmentation of architectural constructions, either for automating production, as design guides when constructing architectural buildings, or for design improvement when segmentation under constrains.

Our block segmentation algorithm in chapter 3 lowers the amount of material required for production by up to 48%. The problem is NP hard by nature and our method shows a large improvement in material compared to state of the art methods.

The method in the rationalization paper [SNS+] reduces the amount of milling by precutting the model using a wire. We show that up to 98% of the required milling can be removed by this method for models with high Gaussian curvature. This paper also takes into account the robotic constraints in production.

We also introduced three segmentation methods for cutting with hot-blade (elastic) in chapter 5. Each of these methods produce different aesthetic segmentations, which is shown in the final design by the seams in the segmentation. Each method takes into account different aspects of the elastica constraints and the production limitations.

An interactive design tool for elastica segmentation was developed and tested on a workshop for architect students, which lead to the paper [BBC+] where the challenges and possible improvements of the interactive tool was discussed.

All in all, there are is a big future for segmentation and rationalization both when considering the price, the aesthetic design guides, and overcoming production constraints.

Bibliography

[BAS14] J. A. Bærentzen, R. Abdrashitov, and K. Singh. Interactive shape modeling using a skeleton-mesh co-representation. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, 33(4), 2014.

[BBaE+] David Brander, Andreas Bærentze, anton Evgrafov, Jens Gravesen, Steen Markvorsen, Toke B Nørbjerg, Peter Nørtoft, and Kasper H. Steenstrup. Hot blade cuttings for the building industries. Unpublish paper waiting for accpetes.

[BBC+] David Brander, Andreas Bærentze, Kenn Clausen, Ann-Sofie Fisker, Jens Gravesen, Morten N. Lund, Toke B. Nørbjerg, and Kasper Steenstrup Asbjørn Søndergaard. Designing for robotic hot-blade cutting. Unpublish paper waiting for accpetes in AAG 2016.

[BGAA12] J Andreas Bærentzen, Jens Gravesen, François Anton, and Henrik Aanæs. *Guide to computational geometry processing: foundations, algorithms, and methods*. Springer Science & Business Media, 2012.

[BGN15] David Brander, Jens Gravesen, and Toke Bjerge Nørbjerg. Approximation by planar elastic curves. *arXiv preprint arXiv:1509.00703*, 2015.

[BHP99] Gill Barequet and Sariel Har-Peled. E ciently approximating the minimum-volume bounding box of a point set in three dimensions. In *Proc. 10th ACMSIAM Sympos. Discrete Algorithms*, pages 82–91. Citeseer, 1999.

[BLM10] Ernesto G Birgin, Rafael D Lobato, and Reinaldo Morabito. An effective recursive partitioning approach for the packing of identical rectangles in a rectangle. *Journal of the Operational Research Society*, pages 306–320, 2010.

[Bur16] Mark Burry. Robots at the sagrada família basilica: A brief history of robotised stone-cutting. In *Robotic Fabrication in Architecture, Art and Design 2016*, pages 2–15. Springer, 2016.

[CC06] Chih-Hsing Chu and Jang-Ting Chen. Tool path planning for five-axis flank milling with developable surface approximation. *The International Journal of Advanced Manufacturing Technology*, 29(7-8):707–713, 2006.

[CD15] Lixin Cao and Lei Dong. Positioning method of a cylindrical cutter for ruled surface machining based on minimizing one-sided hausdorff distance. *Chinese Journal of Aeronautics*, 28(5):1564 – 1573, 2015.

[Chi04] John CJ Chiou. Accurate tool position for five-axis ruled surface machining by swept envelope approach. *Computer-Aided Design*, 36(10):967–974, 2004.

[DSG+11] Kathrin Dörfler, Timothy Sandy, Markus Giftthaler, Fabio Gramazio, Matthias Kohler, and Jonas Buchli. Mobile robotic brickwork. 2011.

[EF97] Gershon Elber and Russ Fish. 5-axis freeform surface milling using piecewise ruled surface approximation. *Journal of Manufacturing Science and Engineering*, 119(3):383–387, 1997.

[Elb95] Gershon Elber. Model fabrication using surface layout projection. *Computer-Aided Design*, 27(4):283 – 291, 1995.

[Fer14] Jelle Feringa. Entrepreneurship in architectural robotics: The simultaneity of craft, economics and design. *Architectural Design*, 84(3):60–65, 2014.

[FNI+12] Simon Flöry, Yukie Nagai, Florin Isvoranu, Helmut Pottmann, and Johannes Wallner. *Ruled free forms*. na, 2012.

[FP10] Simon Flöry and Helmut Pottmann. Ruled surfaces for rationalization and design in architecture. *LIFE in: formation. On Responsive Information and Variations in Architecture*, pages 103–109, 2010.

[FS15] Jelle Feringa and Asbjørn Søndergaard. Fabricating architectural volume. *Fabricate*, 2015.

[JKS05] Dan Julius, Vladislav Kraevoy, and Alla Sheffer. D-Charts: Quasi-Developable Mesh Segmentation. *Computer Graphics Forum*, 2005.

[JTT+14] Caigui Jiang, Chengcheng Tang, Marko Tomicic, Johannes Wallner, and Helmut Pottmann. Interactive modeling of architectural freeform structures - combining geometry with fabrication and statics. In P. Block, J. Knippers, and W. Wang, editors, *Advances in Architectural Geometry*. Springer, 2014.

[KGW14] F Kohler, M Gramazio, and J Willmann. The robotic touch: How robots change architecture, 2014.

[Lev09] Raphael Linus Levien. From spiral to spline: Optimal techniques in interactive curve design. 2009.

[LSVT15] Marco Livesu, Alla Sheffer, Nicholas Vining, and Marco Tarini. Practical hex-mesh optimization via edge-cone rectification. *Transactions on Graphics (Proc. SIGGRAPH 2015)*, 34(4), 2015.

[MFS13] Wes McGee, Jelle Feringa, and Asbjørn Søndergaard. *Rob | Arch 2012: Robotic Fabrication in Architecture, Art, and Design*, chapter Processes for an Architecture of Volume, pages 62–71. Springer Vienna, Vienna, 2013.

[MPV00] Silvano Martello, David Pisinger, and Daniele Vigo. The three-dimensional bin packing problem. *Operations Research*, 48(2):256–267, 2000.

[MVLS14] Chongyang Ma, Nicholas Vining, Sylvain Lefebvre, and Alla Sheffer. Game level layout from design specification. In *Eurographics 2014*, pages 95–104, 2014.

[Pis05] David Pisinger. Where are the hard knapsack problems? *Computers & Operations Research*, 32(9):2271 – 2284, 2005.

[Pre10] Andrew N Pressley. *Elementary differential geometry*. Springer Science & Business Media, 2010.

[SBB+16] Gregor Steinhagen, Johannes Braumann, Jan Brüninghaus, Matthias Neuhaus, Sigrid Brell-Cokcan, and Bernd Kuhlenkötter. Path planning for robotic artistic stone surface production. In *Robotic Fabrication in Architecture, Art and Design 2016*, pages 122–135. Springer, 2016.

[SBSG] Kasper H. Steenstrup, Andreas Bærentze, Alla Sheffer, and Jens Gavensen. Block segmentation. Manuscript for SIGGRAPH asia 2016.

[SDW+16] Peng Song, Bailin Deng, Ziqi Wang, Zhichao Dong, Wei Li, Chi-Wing Fu, and Ligang Liu. CofiFab: Coarse-to-fine fabrication of large 3d objects. *ACM Transactions on Graphics (SIGGRAPH 2016)*, 2016.

[SFLF15] Peng Song, Zhongqi Fu, Ligang Liu, and Chi-Wing Fu. Printing 3d objects with interlocking parts. *Computer Aided Geometric design (Proc. of GMP 2015)*, 35-36:137–148, 2015.

[SFN+16] Asbjørn Søndergaard, Jelle Feringa, Toke Nørbjerg, Kasper Steenstrup, David Brander, Jens Graversen, Steen Markvorsen, Andreas Bærentzen, Kiril Petkov, Jesper Hattel, et al. Robotic hot-blade cutting. In *Robotic Fabrication in Architecture, Art and Design 2016*, pages 150–164. Springer, 2016.

[SNS+] Kasper H. Steenstrup, Toke B. Nørbjerg, Asbjørn Søndergaard, J. Andreas Bærentzen, and Jens Gravesen. Cuttable ruled surface strips for milling. Unpublish paper waiting for accpetes in AAG 2016.

[Søn14] Asbjørn Søndergaard. Odico formwork robotics. *Architectural Design*, 84(3):66–67, 2014.

[TSG+15] Chengcheng Tang, Xiang Sun, Alexandra Gomes, Johannes Wallner, and Helmut Pottmann. Form-finding with polyhedral meshes made simple. In *ACM SIGGRAPH 2015 Posters*, page 5. ACM, 2015.

[WB05] Andreas Wächter and T. Lorenz Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2005.

[WE14] Charlie CL Wang and Gershon Elber. Multi-dimensional dynamic programming in ruled surface fitting. *Computer-Aided Design*, 51:39–49, 2014.

[YIO+15] Hironori Yoshida, Takeo Igarashi, Yusuke Obuchi, Yosuke Takami, Jun Sato, Mika Araki, Masaaki Miki, Kosuke Nagata, Kazuhide Sakai, and Syunsuke Igarashi. Architecture-scale human-assisted additive manufacturing. *ACM Transactions on Graphics (TOG)*, 34(4):88, 2015.

[Zho10] Yayun Zhou. *Optimization with Ruled Surface*. Logos Verlag Berlin GmbH, 2010.

[ZLAK14] Henrik Zimmer, Florent Lafarge, Pierre Alliez, and Leif Kobbelt. Zometool shape approximation. *Graphical Models*, 76(5):390–401, 2014.

Hot Blade Cuttings for the Building Industries

David Brander, Andreas Bærentzen, Anton Evgrafov, Jens Gravesen, Steen Markvorsen, Toke Bjerge Nørbjerg, Peter Nørtoft, and Kasper Steenstrup

Abstract The constructions of advanced architectural designs are presently very labour intensive, time consuming, and expensive. They are therefore only applied to a few prestige projects, and it is a major challenge for the building industry to bring the costs down and thereby offer the architects more variability in the (economically allowed) designs - i.e., to allow them to think out of the box. To address this challenge The Danish National Advanced Technology Foundation (now InnovationsFonden) is currently supporting the BladeRunner project that involves several Danish companies and public institutions. The project aims to reduce the amount of manual labour as well as production time by applying robots to cut expanded polystyrene (EPS) moulds for the concrete to form doubly curved surfaces. The scheme is based upon the so-called Hot Wire or Hot Blade technology where the surfaces are essentially swept out by driving an Euler elastica through a block of EPS. This paper will be centered around the mathematical challenges encountered in the implementation of this idea. They are mainly concerned with the rationalization of the architects' CAD drawings into surfaces that can be created via this particular sweeping and cutting technology.

1 The need for low cost procedures

A recurring theme in the architectural industry of today is a conflict between the design ambitions of the architect and the economic realities of fabrication processes.

David Brander, DTU Compute, e-mail: dbra@dtu.dk
Andreas Bærentzen, DTU Compute, e-mail: janba@dtu.dk
Anton Evgrafov, NTNU, Dept. of Math., e-mail: anton.evgrafov@math.ntnu.no
Jens Gravesen, DTU Compute, e-mail: jgra@dtu.dk
Steen Markvorsen, DTU Compute, e-mail: stema@dtu.dk
Toke Bjerge Nørbjerg, DTU Compute, e-mail: tono@dtu.dk
Peter Nørtoft, AutoDesk, Denmark, e-mail: penn@dtu.dk
Kasper Steenstrup, DTU Compute, e-mail: khor@dtu.dk

The desire to create unique and attractive designs, often motivated by the competitive industry climate, leads to the use of curved geometries and bespoke elements that can be conceived easily within modern CAD systems, but, in reality, are prohibitively expensive to build. This results in compromises at the so-called *rationalization* stage, where the design is adjusted within an engineering context for production purposes. A typical example is where the desired shape of a building leads to panels (or some other element) of perhaps 200 different shapes. Consultation with fabrication contractors then reveals that dramatic cost reductions can be achieved if the design is adjusted so that only 20 unique elements are used, with repetition, instead of the 200. Finally, budgetary considerations force a compromise of the original design.

The present project addresses this issue, in particular within the domain of the production of formwork for concrete constructions. The shape of the surface of a facade or other element is produced in negative in several pieces of expanded polystyrene (EPS) foam that is used as a mould for concrete casting. EPS can also be used in positive shape production for some applications by applying a coating and retaining the EPS as a structural element. For curved surfaces the currently available technology for shaping the EPS is computer numerically controlled milling, a slow, and therefore expensive, process. The *BladeRunner* project, supported by the Innovation Fund Denmark, is presently developing new processes, *robotic Hot Wire/Blade cutting*, for carving shapes out of EPS using a robotically controlled heated wire or blade. The technology is projected to reduce production time of architectural formwork by a factor of over 100, and to bring the cost of production for advanced shapes into the domain of financial feasibility.

Fig. 1: Robotic Hot Wire cutting in Odense, Denmark.

2 Principles of Hot Blade cuttings

The essential principle of both Hot Wire and Hot Blade cutting is very simple. A heated wire or blade, either of which we may think of as a "blade", is moved relative to a block of EPS, carving out a surface through the block (Figure 1). Either the block or the blade, or both, are controlled by a robot. For definiteness, we regard the block as fixed and the blade as moving.

2.1 Hot wire cutting and its limitations

For the wire technology, the wire is held tight, forming a straight line, and thus sweeps out a *ruled surface*. This technology is limited in its ability to approximate general freeform surfaces. This can be seen by considering the *Gaussian curvature* of a surface, which is defined as follows: through any point p on a surface a curve is obtained by intersecting the surface with a plane perpendicular to the tangent plane at that point (Figure 2). The *normal curvature* associated to the tangent direction of this curve is the curvature of this curve at the point p, with the sign determined by a fixed choice of surface normal vector (Figure 2, left). The maximum and minimum values obtained from all possible tangent directions at p are called the *principal curvatures*, κ_1 and κ_2, and their product is the Gaussian curvature $K = \kappa_1 \kappa_2$. In the saddle surface shown at Figure 2, the principal curves bend in opposite directions away from the tangent plane and so κ_1 and κ_2 have opposite signs and $K < 0$.

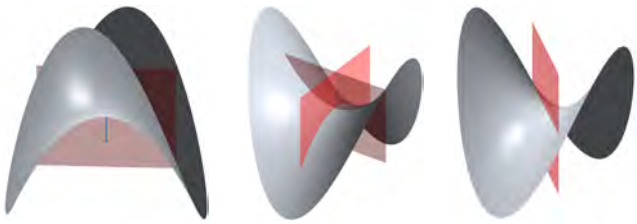


Fig. 2: A saddle surface. Left: the normal curvature defined by this intersection curve is positive if the downward pointing surface normal is chosen. Middle: the planes defining the principal curvatures at the center. Right: this normal section is a straight line; the normal curvature is zero in this direction.

If the Gaussian curvature is positive, then κ_1 and κ_2 at p have the same sign, and any other tangent direction at p has normal curvature κ_n with $\kappa_1 \geq \kappa_n \geq \kappa_2$. Therefore κ_n cannot be zero in this case. Now for an arbitrary arc-length parameterized curve γ in the surface the acceleration vector decomposes as $\gamma''(s) = \kappa_g(s)\nu(s) + \kappa_n(s)N(s)$, where N is the surface normal, and κ_n is the normal curva-

ture in the direction of γ'. It follows that, if $\kappa_n \neq 0$, then the acceleration is non-zero and thus the curve cannot be a straight line. On the other hand, a ruled surface is defined to be a surface swept out by a smoothly varying family of straight lines: through every point of a ruled surface there is a straight line lying in the surface. Therefore, by the discussion above, a ruled surface cannot have positive Gaussian curvature; moreover, there is no chance of obtaining a good local approximation for a positively curved surface by a ruled surface (Figure 3).

Fig. 3: At a point of positive Gaussian curvature the surface is bowl-shaped. No straight line tangent to the surface can approximate a curve in the surface.

Figure 4 shows (center) an approximation of a negatively curved surface by ruled strips. The ruling directions are chosen to be close to the *asymptotic directions*, namely directions where the normal curvature is zero. However, even for negatively curved surfaces, it is in general not possible to obtain a tangent continuous approximation - the tangent planes do not match along adjacent strips.

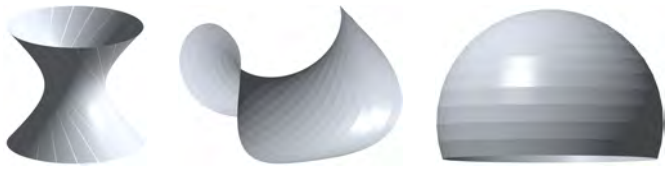


Fig. 4: Left to right: The hyperboloid (a ruled surface); Approximation of a negatively curved surface by *strips* of ruled surfaces; Ruled strip approximation of a positively curved surface.

2.2 Hot blade cutting

The blade concept is much more general: the end points and the *tangents* at the endpoints vary during the sweeping. We will assume that the curve lies in a plane, that is, that the end tangents are co-planar. This restriction makes the mechanical implementation of the process easier, both in terms of choosing the cross-sectional

shape of the blade design and allowing for the possibility that only one edge of the physical blade is heated.

An elastic rod, of a fixed length and with end points and end tangents at a given position, assumes the shape of an Euler elastica (discussed below). These curves are well understood mathematically and are given in terms of elliptic functions.

In order to apply either of these technologies to a given CAD design, a *rationalization* of the relevant surface is necessary: the surface must first be *segmented* into pieces, each of which can be approximated within a given tolerance by a surface swept out by curves of the relevant type (lines or a family of elastica). Next, each segment is foliated by curves each of which is approximated by a curve of the type in question. Finally, the data for producing these curve sweepings is given to the robot control software.

Methods for rationalization for Hot Wire cutting have been given already in the literature (see below). Therefore, in this article, the rationalization project we are concerned with consists of both developing a segmentation algorithm for blade-cut surfaces, and an algorithm for approximating arbitrary spline curves by Euler elastica.

2.3 Previous related works

Pottmann and Flörey [7] developed a ruled surfaces segmentation algorithm using the fact that on ruled surfaces one of the asymptotic directions at a point must be tangent to the ruling, giving natural candidates for the ruling direction in the surface to be approximated. As such, this segmentation strategy does not generalize to the case of hot *blade* cutting, therefore a new strategy must be developed.

For the Hot Blade technology, some work has been done in the late 1990's to the early 2000's by a group at Delft: see [8, 4] and associated references. The use of the Hot Blade technology there is somewhat different, as the aim is to produce 3-dimensional solid rapid prototype models from EPS via a so-called "thick-layered fabrication" process. The solid is built up by stacking many thick slices, and the curved surface that needs to be cut is only a narrow strip around the boundary of each slice. Therefore, the segmentation problem is completely different from the surface segmentation problem that will apply to the BladeRunner process.

The work of the Delft group is concentrated on approximating the blade shape and algorithms for tool positioning. The approach they use for approximating the blade shape is to apply a numerical method to minimize the bending energy. Below we will use a different approach that takes advantage of the known analytic solutions for this problem to give an explicit parameterization of the space of solutions. This allows us to move easily in the space of solutions, calculate gradients, and use standard optimization packages to find an elastic curve that approximates an arbitrary given curve.

3 The Euler elastica

We describe here a parameterization of the space of planar elastic curve segments. More details of this parameterization and further references can be found in [3]. An introduction to the theory of elastic curves, with historical references, can be found in [11]. Other works on the topic of elastic curves as splines are [1, 2, 5, 6].

3.1 The Euler-Lagrange equation

We give here a brief derivation of the differential equation that determines the solutions to the elastica problem. The reader unfamiliar with the calculus of variations could take this derivation for granted and proceed directly to the solutions given in the next subsection. Let $\gamma : [0,\ell] \to \mathbb{R}^2$ be a plane curve segment parameterized by arc-length, and define an angle function $\theta(s)$ by $\dot{\gamma}(s) = (\cos\theta(s), \sin\theta(s))$. A curve segment of length ℓ starting at (x_0, y_0) and ending at (x_ℓ, y_ℓ) satisfies

$$x_\ell = x_0 + \int_0^\ell \cos\theta \, \mathrm{d}s\,, \quad y_\ell = y_0 + \int_0^\ell \sin\theta \, \mathrm{d}s\,. \tag{1}$$

Let κ denote the curvature $\theta'(s)$. An *elastica* is a curve that minimizes the *bending energy*

$$\frac{1}{2}\int_0^\ell \kappa(s)^2 \, \mathrm{d}s\,. \tag{2}$$

The equations defining the elastica are obtained from a variational problem: suppose γ is an elastica from (x_0, y_0) to (x_ℓ, y_ℓ) with angle function $\theta(s)$. A smooth variation is given by the family γ_t with angle function $\theta_t(s) = \theta(s) + t\psi(s)$, where ψ is a differentiable function with $\psi(0) = \psi(\ell) = 0$. Applying the method of Lagrange multipliers we find that, if γ minimizes the energy among such curves, then the angle function θ satisfies:

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}s^2} + \lambda_1 \sin\theta - \lambda_2 \cos\theta = 0\,. \tag{3}$$

Setting $(\lambda_1, \lambda_2) = \lambda(\cos\phi, \sin\phi)$, with $\lambda \geq 0$, this becomes $\frac{\mathrm{d}^2\theta}{\mathrm{d}s^2} + \lambda \sin(\theta - \phi) = 0$. Note that $\lambda = 0$ if and only if κ is constant, i.e the curve γ is either a straight line segment or a piece of a circle. If $\lambda \neq 0$, set

$$\widetilde{\gamma}(s) = \sqrt{\lambda} R_{-\phi} \gamma\left(\frac{s}{\sqrt{\lambda}}\right), \qquad R_\phi = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}. \tag{4}$$

Then $\tilde{\gamma}$ is a scaled and rotated version of γ and thus also an elastica. Its tangent angle is $\tilde{\theta}(s) = \theta(s/\sqrt{\lambda}) - \phi$ and it satisfies the normalized pendulum equation $\frac{\mathrm{d}^2\tilde{\theta}}{\mathrm{d}s^2} = -\sin\tilde{\theta}$. Hence we conclude that, up to a scaling and rotation of the ambient space,

all arc-length parameterized elastica $\gamma : [0,1] \to \mathbb{R}^2$, with non-constant curvature κ, can be expressed as:

$$\gamma(s) = \gamma(0) + \int_0^s (\cos\theta(t), \sin\theta(t))\,\mathrm{d}t \tag{5}$$

where

$$\ddot{\theta} = -\sin\theta\,. \tag{6}$$

3.2 The space of elastic curve segments

We now find some suitable parameters to describe the space of elastic curve segments. First, it is well known that the solutions to (6) can be expressed in closed form via the elliptic functions sn, cn, and dn. These solutions can be found in Love [12]. There are two classes of solution curves: those with inflection points (i.e. points where $\kappa = \dot{\theta} = 0$) and those without inflections. Each class of solutions is a 1-parameter family.

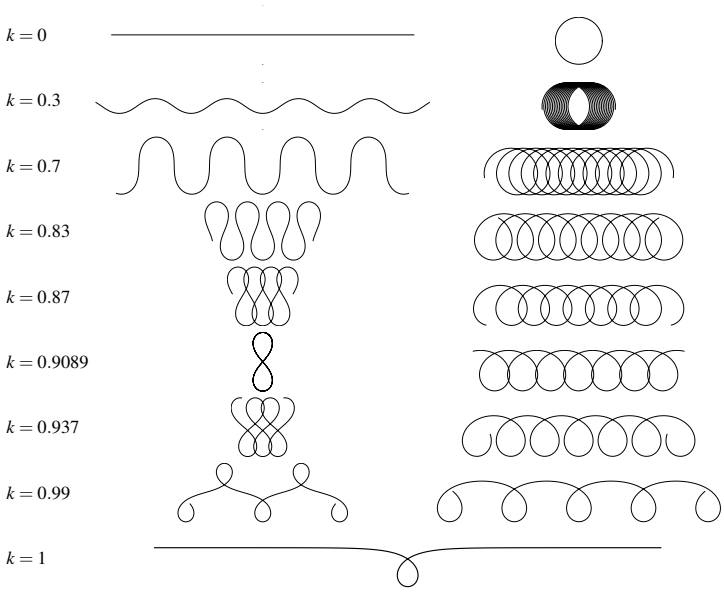


Fig. 5: Euler elastica. Left: inflectional. Right: non-inflectional. The respective elastica – with values of k ranging from 0 at the top to 1 at the bottom – are plotted.

The *inflectional elastica* starting at $(0,0)$ with initial angle $\theta(0)=0$ and $\dot{\theta}(0)>0$ is

$$\zeta_k(s)=\zeta(s,k)=\begin{pmatrix}2E(s,k)-s\\ 2k(1-\mathrm{cn}(s,k))\end{pmatrix},\qquad \text{where } k=\dot{\theta}(0)/2.$$

A *segment* of such a curve is determined by the value k, a starting point s_0 and a length ℓ. Finally, adding a scaling S, a rotation ϕ and a translation (x_0,y_0), we have a standard representation $\gamma_{(k,s_0,L,S,\phi,x_0,y_0)}:[0,1]\to\mathbb{R}^2$ for a segment of an inflectional elastic curve:

$$\begin{aligned}\gamma_{(k,s_0,\ell,S,\phi,x_0,y_0)}(t)&=SR_\phi\left(\zeta_k(s_0+\ell t)\right)+\begin{pmatrix}x_0\\ y_0\end{pmatrix}\\ &=SR_\phi\begin{pmatrix}2E(s_0+\ell t,k)-(s_0+\ell t)\\ 2k(1-\mathrm{cn}(s_0+\ell t,k))\end{pmatrix}+\begin{pmatrix}x_0\\ y_0\end{pmatrix},\end{aligned}$$

where

$$E(s,k):=\int_0^s \mathrm{dn}^2(\tau,k)\mathrm{d}\tau.$$

Note that the arc-length parameter in this representation is $s=S(s_0+\ell t)$ and not t and that the length is $L=S\ell$.

Similarly, we obtain a standard representation of a *non-inflectional* elastic curve segment:

$$\gamma_{(k,s_0,\ell,S,\phi,x_0,y_0)}(t)=SR_\phi\begin{pmatrix}(1-\frac{2}{k^2})(s_0+\ell t)+\frac{2}{k}E\left(\frac{s_0+\ell t}{k},k\right)\\ \frac{2}{k}\left(1-\mathrm{dn}\left(\frac{s_0+\ell t}{k},k\right)\right)\end{pmatrix}+\begin{pmatrix}x_0\\ y_0\end{pmatrix}.$$

4 Sweeping surfaces with Euler elastica

The figures in this section illustrate examples of surfaces foliated by continuously varying segments of Euler elastica. These examples are constructed by parameterizing the space of planar elastica segments, as in the previous section, choosing a small number of sample curve segments, and then interpolating the data through the parameter space. Hence each surface is swept by a family of planar elastica.

In principle, all of these surfaces could be produced by robotic hot-blade cutting, but there are technical issues that depend on the practical implementation. For example, the surface on the left in Figure 7 is a surface of revolution, but one end of the profile curve is much closer to the axis of rotation than the other. This means that the blade moves much more slowly on the inner end resulting in too much melting of the EPS. One solution is to segment the surface into several pieces, cut separately. Another is to approximate this surface by some other, non-rotational, family of elastic segments.

Yet another restriction arises if a flat blade is used, rather than a cylindrical rod (see Figure 8). With the flat blade design, the blade is curved in a plane perpendic-

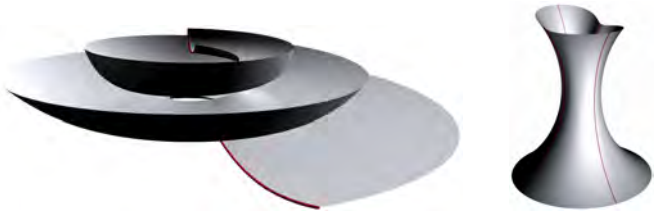


Fig. 6: Examples of surfaces swept by continuously varying elastic curve segments.

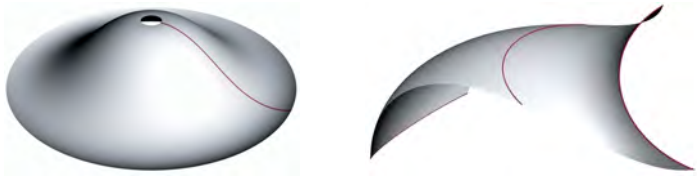


Fig. 7: Two technically problematic situations.

ular to the plane of the blade. If one edge of the blade is heated, the motion of the blade should be roughly in the direction of this edge, that is, approximately perpendicular to the plane of the curve, in order to cut a path through the material. Another way to say this is that the elastic curves should be as close as possible to geodesic curves on the surface under construction. To require that these planar elastic curves are true geodesics would place too large a restriction on the uses of this method; so we apply a tolerance instead. Both surfaces shown in Figure 6 are reasonable candidates for cutting with a flat blade. The second surface in Figure 7 however, would be impractical with the given elastica foliation. The osculating plane of the elastic curve shown is very close to the tangent plane of the surface; thus the hot edge is pointing out of the surface, and the blade would not be able to progress in the required direction.

5 Approximation by Euler elastica

In this section we consider the problem of approximating a given planar spline curve $\mathbf{x} : [0,1] \to \mathbb{R}^2$ by a planar elastic curve. We present two different approaches to this problem. In the first we try to find the parameters $(k, s_0, \ell, S, \phi, x_0, y_0)$ of the elastica that has the best fit to the curve $\mathbf{x}$. This is a nonlinear optimization problem, and the final result depends crucially on a good initial guess. The second approach is purely numerical – we model the elastica with a spline on a much finer knot vector than

Fig. 8: Left: A cylindrical rod can cut in any direction that is close to perpendicular to the tangent of the curve. Right: A flat (or ribbon) blade design (with its hot edge indicated in red) moves well only in the directions close to the direction of the cutting edge.

the original curve, and then solve a constrained optimization problem minimizing the elastic energy under the constraint of being within some distance to the original curve.

5.1 Analytic approach: finding the parameters for the elastica

We describe here the essentials of the gradient driven analytic approach. For full details, see the article [3].

We wish to find the elastic curve segment which most closely resembles the given spline curve $\mathbf{x}$. We choose to minimize the L_2-distance between the curves. For a given set of parameters, the elastic curve segment $\gamma_{(k,s_0,\ell,S,\phi,x_0,y_0)}$ is parameterized with constant speed ℓS over the interval $[0,1]$. The spline curve is also defined on $[0,1]$, but its speed is not necessarily constant. Since the L_2-norm compares points at corresponding parameter values, we need to reparameterize either the spline or the elastica for the L_2-distance to be a good measure of the curves' resemblance. The simplest way is to reparameterize the elastica using the arc length s of the spline which can be calculated as

$$s(t) = \int_0^t \frac{\mathrm{d}s}{\mathrm{d}t}\,\mathrm{d}\tau = \int_0^t \|\mathbf{x}'(\tau)\|\,\mathrm{d}\tau, \tag{7}$$

and the length of the spline is then $L = s(1)$. We now consider the minimization problem

$$\text{minimize}_{k,s_0,\ell,S,\phi,x_0,y_0}\, \mathscr{E}(k,s_0,\ell,S,\phi,x_0,y_0), \tag{8}$$

where

$$\mathscr{E} = \frac{1}{2}\int_0^1 \left\|\mathbf{x}(t) - \gamma_{(k,s_0,\ell,S,\phi,x_0,y_0)}(s(t)/L)\right\|^2 \|\mathbf{x}'(t)\|\,\mathrm{d}t \tag{9}$$

is the square of the L_2-distance between the spline and the elastica segment.

We use a gradient driven optimization package IPOPT [9], so we need the partial derivatives of the objective function $\mathscr{E}$ with respect to the parameters $(c_1,\ldots,c_7) = (k,s_0,\ell,S,\phi,x_0,y_0)$, i.e.,

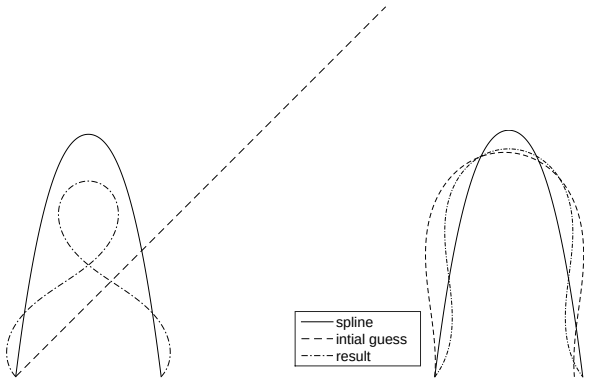


Fig. 9: Approximating a spline by elastica. To the left an arbitrary (bad) initial guess and to the right our guess. The solid line is the spline, the dashed curves are the initial guess, the dash-dotted curves are the optimized approximations.

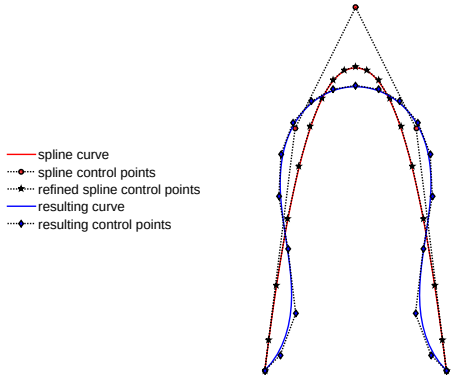


Fig. 10: A spline approximation of an elastica (blue) constrained by a target spline (red). The end positions and tangents have been fixed.

$$\frac{\partial \mathscr{E}}{\partial c_i} = -\int_0^1 \left\langle \frac{\partial \gamma_{\mathbf{c}}}{\partial c_i}(s(t)/L), \mathbf{x}(t) - \gamma_{\mathbf{c}}(s(t)/L) \right\rangle \|\mathbf{x}'(t)\| \, \mathrm{d}t \,. \tag{10}$$

The optimization problem is non-convex, so there are several local minima for $\mathscr{E}$. Therefore the optimization gives different results depending on the initial values of the parameters, cf. Figure 9. It is therefore necessary for us to have a good initial guess. We describe next our method for finding an initial guess. The full details can be found in [3].

We find the initial guess by considering the differential equation (3). If we let $u = \frac{1}{\lambda}(\lambda_2 x - \lambda_1 y)$ then the differential equation can be written as $\frac{\mathrm{d}^2\theta}{\mathrm{d}s^2} = \lambda \frac{\mathrm{d}u}{\mathrm{d}s}$, and integrating this yields

$$\kappa = \frac{\mathrm{d}\theta}{\mathrm{d}s} = \lambda\, u + \alpha = \lambda_2 x - \lambda_1 y + \alpha\,. \tag{11}$$

Letting θ_u denote the angle between the u-axis and the tangent, we have

$$\cos\theta_u = \frac{1}{\lambda}\begin{pmatrix}\lambda_2\\ -\lambda_1\end{pmatrix}\cdot\begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix} = \frac{\mathrm{d}u}{\mathrm{d}s},$$

so

$$\frac{\mathrm{d}\sin\theta_u}{\mathrm{d}u} = \frac{\mathrm{d}s}{\mathrm{d}u}\frac{\mathrm{d}\sin\theta_u}{\mathrm{d}s} = \frac{1}{\cos\theta_u}\cos\theta_u\frac{\mathrm{d}\theta_u}{\mathrm{d}s} = \kappa = \lambda u + \alpha,$$

and thus

$$\sin\theta_u = \frac{1}{2}\lambda\, u^2 + \alpha\, u + \beta\,.$$

As $(\lambda_1, \lambda_2) = S^{-2}(\cos\phi, \sin\phi)$ we get estimates for the scale S and the angle ϕ by solving the first equation with respect to $\lambda_1, \lambda_2, \alpha$ in the least square sense. In a similar manner we can estimate β, and by analysing the resulting parabola we can determine whether we should use an elastica with or without inflections and estimate the parameter k. In the next step we determine which segment of the elastica we should use, i.e., estimate s_0 and ℓ. We finally determine the translation (x_0, y_0) by a least square fit. If we want end point interpolation then we can achieve that by a final scaling, rotation, and translation.

5.2 A purely numerical approach

We have described a method for approximating a spline curve $\mathbf{x} : [0,1] \to \mathbb{R}^2$ by a segment of an elastic curve, represented by an analytic solution in terms of elliptic functions. An alternative approach is to approximate the spline by another spline curve $\mathbf{y}$ which is intended to be close to an elastica, in the sense that it minimizes the elastic energy. This approach could be advantageous for practical reasons. For example, existing CAD software and other mathematical software and algorithms already work with the data structure of splines.

We will use a refined knot vector for the new spline curve $\mathbf{y}$. By knot insertion we express both the target spline $\mathbf{x}$ and the elastica approximation $\mathbf{y}$ using the same basis functions (B-splines). This gives us control points $\mathbf{x}_i$ and $\mathbf{y}_i$, $i = 1, \ldots, n$, that we can compare. We now seek to minimize the bending energy (2) of the new spline curve $\mathbf{y}$, with control points $\mathbf{y}_i$, while staying close to the original curve $\mathbf{x}$, with control points $\mathbf{x}_i$. The difference between these spline curves is also a spline curve, with control points $\mathbf{x}_i - \mathbf{y}_i$, and the distance between the curves is captured by the distance between the control points. These points have coordinates $(\mathbf{x}_i - \mathbf{y}_i)\cdot\mathbf{e}_i$,

where $\mathbf{e}_1 = (1,0)$ and $\mathbf{e}_2 = (0,2)$. That is, we consider the constrained optimization problem:

$$\text{minimize}_{\mathbf{y}_1,\ldots,\mathbf{y}_n} \qquad \frac{1}{2}\int_0^1 \kappa_{\mathbf{y}}^2 \,\mathrm{d}s\,, \tag{12}$$

$$\text{such that} \qquad -\varepsilon \leq \left(\mathbf{x}_i - \mathbf{y}_i\right)\cdot \mathbf{e}_j \leq \varepsilon\,, \qquad i = 1,\ldots,n, j = 1,2\,. \tag{13}$$

We need to constrain the problem additionally, e.g., by fixing the positions and tangents at the two end points. On top of this, we thus have an optimization, or sampling, over end points and tangents. For end point interpolation we simply put $\mathbf{y}_1 = \mathbf{x}_1$, $\mathbf{y}_n = \mathbf{x}_n$, and remove these two control points from both the optimization and the constraints. The tangent constraints just specify directions along which $\mathbf{y}_2$ and $\mathbf{y}_{n-1}$ can move. The length could also be specified. In any case, we are no longer looking for the elastica that minimizes the distance to $\mathbf{x}$, but rather for an elastica that is ε-close $\mathbf{x}$. If none of the constraints are active at the end of the optimization we conclude that we have obtained an elastica which is closer to the target spline than ε. This is of course only true up to the discretization error resulting from using splines to model elastica. By refining the knot vector we obtain a smaller discretization error, and we can validate the solution by checking the differential equation (11). An example of this approach is shown in Figure 10.

A disadvantage of this method is that we cannot guarantee that our solution $\mathbf{y}$ is close to an elastica – only that it has less bending energy than the input curve $\mathbf{x}$. For this reason, we have chosen to work with the analytic approach outlined in the previous subsection.

6 Surface rationalization

Before a CAD surface can be realized as a mould in the form of a collection of EPS blocks it needs to be divided into patches. Each individual patch is approximated by a surface swept by a hot blade, i.e., a surface foliated by planar elastic curves.

In fact, we need to consider two processes: *blocking* and *segmentation*. Blocking is the process of dividing a surface into blocks such that each block can be cut individually using either a hot wire or a hot blade. Segmentation on the other hand is the process of dividing the surface into patches swept by elastica or ruled patches. If blocking is performed before segmentation, we simply divide the 3D shape into blocks and then fit the best possible ruled or elastic surface patch to each block - possibly taking constraints between block boundaries into account. Doing segmentation first is arguably harder, but has certain benefits: knowing which segments intersect a given block can be used to inform the blocking procedure.

In the following, we consider a more concrete approach to segmentation in the context where we assume that blocking has been performed first.

We first consider the problem of approximating a single surface by a surface foliated by planar elastic curves. One way to accomplish this is first to foliate the surface

by planar curves and then approximate these planar curves by elastica. That is, we sweep a plane through the surface, and thereby foliate the surface by planar curves. We then pick a finite number of these planes, approximate each of the corresponding planar sections with a segment of an elastica, calculate endpoints, end tangents, and lengths and interpolate this data to obtain an approximation of the original surface.

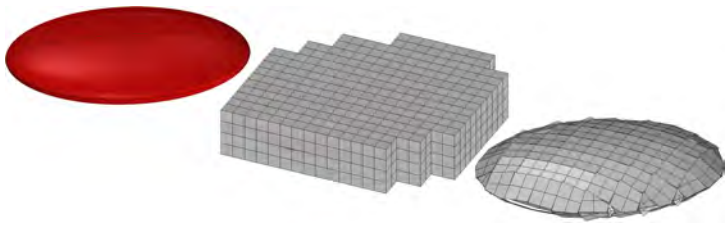


Fig. 11: A simple approach to the rationalization of the red ellipsoid by planar surfaces. No boundary conditions are enforced in this rationalization.

For the general case we imagine our CAD surface sitting inside a collection of EPS blocks. This divides the surface into a collection of pieces each of which is the intersection between a block and the full surface, see figure 11. We now approximate each of these pieces by an elastica swept surface while demanding that neighbouring surfaces fit together in a C^1 fashion. This can be a large global optimization problem and at the end we check to see if the result is within the required tolerance. We then pick the blocks where the tolerance is exceeded, cut these blocks in half and redo the optimization.

In the more complex approach, where segmentation (of the CAD surface) is performed first, several options can be considered. One way is to fit the largest patch that upholds the tolerance criteria to the surface and remove this part to create a reduced surface. This procedure is repeated until the whole surface is removed, i.e., the original surface is covered by patches. Another approach is a patch-growing algorithm as in [10]: A number of patches grow on the surface and whenever two patches meet a competition determines the boundary between the patches. The determining force in the competition is the improvement on the Euler elastica sweep approximation, i.e., the resulting boundary is the one with the largest combined improvement.

For fabrication, each patch needs to be divided into blocks, and this can be difficult on the boundaries; either the blocks need to be cut smaller to align with the boundaries or multiple elastica sweeps are needed, i.e., the block can be cut more than once by the blade.

A third option is a hybrid of the above mentioned methods, where the knowledge from the patch methods guides the placement of the blocks.

7 Examples

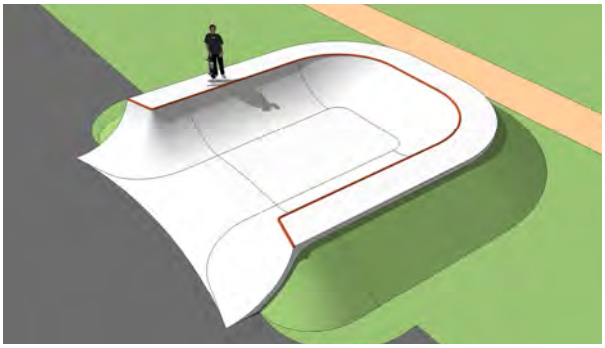


Fig. 12: The skater ramp example.

To illustrate the procedures, we consider the modelling and construction of the skater ramp shown in Figure 12. This CAD surface consists of spline surfaces, some of which are doubly curved. The curved surfaces (see Figure 13) are the ones that need special moulds. Here there are three different types: 1) three ruled parts (the "sides"), 2) two corners with negative curvature at the front of the image and 3) two corners with positive curvature at the back. We will approximate each corner by a surface swept by elastica. The ruled parts can be cut either by the hot wire following the rulings or by the hot blade approximating the curved cross section curve by an elastic curve.

For each corner, the control points give rise to a set of planar spline curves which foliate the surface (see Figure 14 left). These curves can be approximated by elastica as described in Section 5.

If the splines are approximated independently, the control parameters for the resulting elastica might differ quite a lot between two adjacent curves. This is because, for a typical (uncomplicated) curve segment, there can be many different elastic curve segments that approximate it quite closely. To avoid large jumps in the control parameters we use the elastica that approximates the first spline curve as the initial guess for the optimization at the next spline, and so forth.

The optimization is performed with constraints: the approximating elastic curve is in each case required to have the same length and the same end points as the original spline curve. The resulting elastic curves can be seen in Figure 14 right.

Our optimization algorithm minimizes the square on the L_2-distance between the spline curve and the elastica, see (9). Table 1 shows some of these distance values.

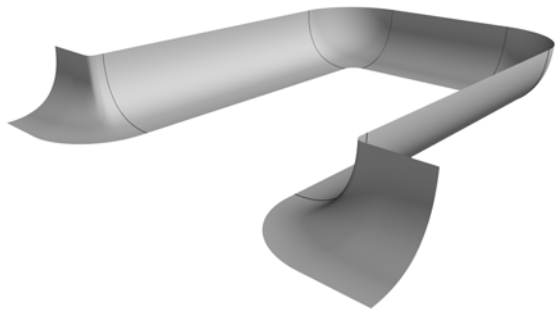


Fig. 13: The spline surfaces of the skater ramp must be approximated by elastica swept surfaces

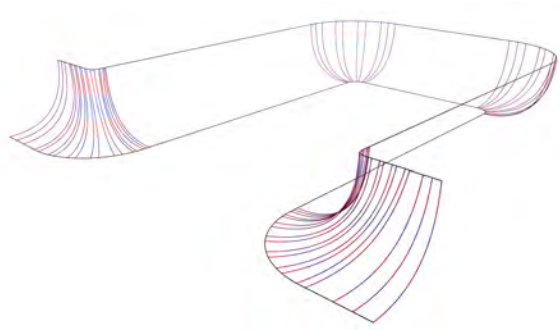


Fig. 14: The corner surfaces are foliated by planar spline curves (blue). Each of these are approximated by an elastic curve (red).

For a visual evaluation of the result, in Figure 15 we have plotted the spline and the approximating elastica in the worst case (i.e. highest L_2 distance). For the corner with negative curvature the curves are nearly indistinguishable. For the positively curved corner, there is clearly a difference, but the overall shape is the same, and the approximation is certainly within any conceivable tolerance for this particular application.

	Min	Max	Average
Negative	0,838846837	0,9107882	0,868163184
Positive	5,738445432	5,788943718	5,778703296

Table 1: The optimized value for the L_2-distance for the two corner types. The minimal value corresponds to the elastic curve which best approximates the spline. The height of the ramp is 854.10 with the lengths of the spline curves varying between 1342.6 and 1459.8.

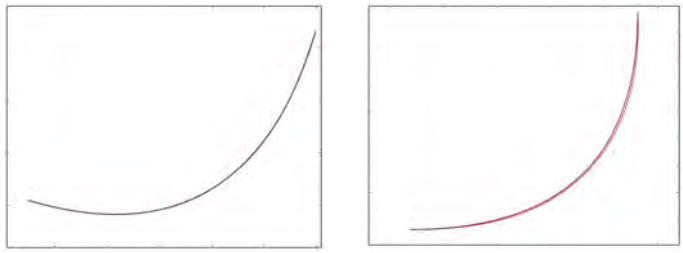


Fig. 15: The original planar spline curve (black) on top of the approximating elastica (red). These are the worst cases for the corners with negative curvature (left) and positive curvature (right).

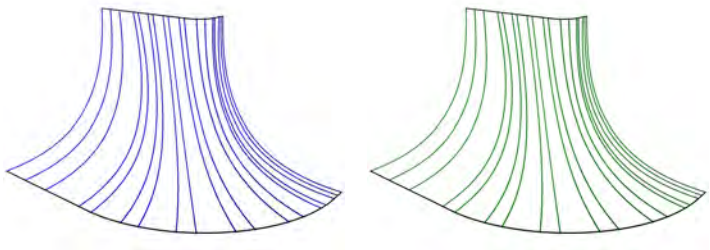


Fig. 16: Left: The surface is foliated by planar spline curves. Right: The surfaces is foliated by elastic curves which approximate the splines on the left figure.

8 Conclusion

Our work on approximating arbitrary spline curves by elastic curves, illustrated here by the test case of the skate ramp, indicate that the problem of approximating most of the CAD surfaces used in architecture by panels of surfaces swept by planar elastica is feasible. The utility of the technology then depends on the technical problem

of designing a blade that can be heated and used to cut EPS in a consistent and predictable way. We will receive input in the near future from project partners who are in the process of developing this blade technology.

Acknowledgements This work was completed with the support of Innovation Fund Denmark, project number 128-2012-3.

References

1. Birkhoff, G., Boor, C.D.: Piecewise polynomial interpolation and approximation. Approximation of Functions, Proc. General Motors Symposium 1964, H.L. Garabedian, ed. pp. 164–190 (1965). Elsevier, Publ. Co. Amsterdam.
2. Borbély, A., Johnson, M.: Elastic splines I: Existence. Constr. Approx. **40**, 189–218 (2014).
3. D. Brander, J. Gravesen, and T. Nørbjerg. Approximation by planar elastic curves. Preprint, arXiv:1509.00703.
4. J.J. Broek, I. Horváth, B. de Smit, A.F. Lennings, Z. Rusák, and J.S.M. Vergeest. Free-form thick layer object manufacturing technology for large-sized physical models. *Automation in Construction*, 11:335–347, 2002.
5. Malcolm, M.: On the computation of nonlinear spline functions. SIAM Journal on Numerical Analysis **14**, 254–282 (1977)
6. Mehlum, E.: Nonlinear splines. Computer aided geometric design (Proc. Conf., Univ. Utah, Salt Lake City, Utah, 1974) pp. 173–207 (1974).
7. S. Flöry and H. Pottmann. Ruled surfaces for rationalization and design in architecture. *Proc. ACADIA*, pages 103–109, 2010.
8. I. Horváth, Z. Kovács, J.S.M. Vergeest, J.J. Broek, and A. de Smit. Free-form cutting of plastic foams: a new functionality for thick-layered fabrication of prototypes. *Proceedings of the TCT'98 Conference, Nottingham*, pages 229–237, 1998.
9. A. Wächter and L.T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program., Ser. A* **106**, pages 25–57, 2006.
10. D. Julius, V. Kraevoy and A. Sheffer. D-charts: Quasi-developable mesh segmentation. *Computer Graphics Forum, Proc. of Eurographics*, pages 581–590, 2005.
11. Levien, R.: From spiral to spline; optimal techniques for interactive curve design. Ph.D. thesis, UC Berkeley (2009)
12. A.E.H. Love, *A treatise on the mathematical theory of elasticity*, Cambridge University Press, 1906.

Robotic Hot-Blade Cutting

An industrial approach to cost-effective production of double curved concrete structures.

Asbjørn Søndergaard[1,*], Jelle Feringa[2], Toke Nørbjerg[3], Kasper Steenstrup[3], David Brander[3], Jens Graversen[3], Steen Markvorsen[3], Andreas Bærentzen[3], Kiril Petkov[4], Jesper Hattel[4], Kenn Clausen[5], Kasper Jensen[5], Lars Knudsen[6] and Jacob Kortbek[6]

1 Odico Formwork Robotics Aps.
asbjorn@odico.dk
[2] Odico Formwork Robotics Aps.
jelle@odico.dk
[3] Technical University of Denmark, dep. of applied mathematics and computer science
{tono, khor, dbra,jgra,stema, janba} @dtu.dk
[4] Technical University of Denmark, dep. of mechanical engineering
{kipekt, jhat} @dtu.dk
[5] GXN A/S
{kec, kgj} @3xn.dk
[6]Danish Technological Institute, Center for Robotics
{lak, jkk} @teknologisk.dk

Abstract. This paper presents a novel method for cost-effective, robotic production of double curved formwork in Expanded Polystyrene (EPS) for in-situ and prefabricated concrete construction. A rationalization and segmentation procedure is developed, which allows for the transliteration of double curved NURBS surfaces to Euler elastica surface segments, while respecting various constraints of production. An 18 axis, tri-robot system approximates double curved NURBS surfaces by means of an elastically deformed and heated blade, mounted on the flanges of two manipulators. Re-orienting or translating either end of the blade dynamically deforms the blade's curvature. The blade follows the contours of the rationalized surface by continuous change in position and orientation of the end-effectors. The concept's potential is studied by a pilot production of a full-scale demonstrator panel assembly.

Keywords: robotic fabrication, Hot-Blade, EPS-molds, cost-efficiency, concrete structures

adfa, p. 3, 2011.

1 Introduction

The vast majority of contemporary building designs are restrained to a formal language of planar surfaces and derivative geometric constructs; a constraint that stems from the practicalities of construction, which favors the use of mass-produced semi-manufactures and – for concrete in particular – modular, reusable formwork systems. An increasing number of high-profile project designs challenge the dominant paradigm. The challenge is posed by advanced building design projects, such as the Kagamigahara Crematorium (Toyo Ito Architects, 2006) and Waalbridge Extension (Zwart & Jansma, under construction), which utilize manual production of formwork to achieve complex curvatures; and building projects which employ large scale CNC-milling to realize advanced structures, such as the Museum Foundation Louis Vuitton by Gehry & Associates (Paris, 2014); the Nordpark cable railway by Zaha Hadid Architects (Nordpark, 2007), the Metz Pompidou by Shigeru Ban (Metz, 2010).

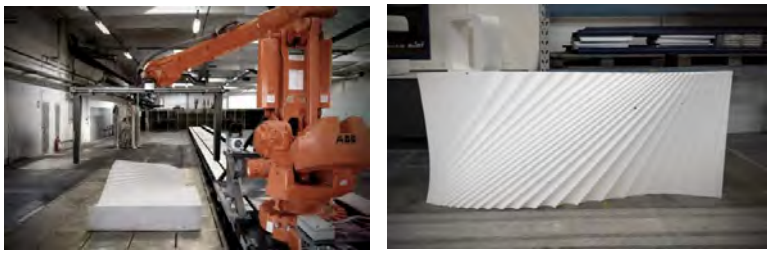


Fig. 1. Large scale RHWC production at Odico (left), and hotwire cut production sample (right)

However, neither manual formwork production, nor large scale CNC-milling provides a cost-effective option for general construction, and projects of this type therefore require extraordinary budget frameworks for realization. Recent developments in architectural robotics by authors of this paper have demonstrated novel, cost-effective means of producing bespoke formwork with the constraint of being limited to ruled surface. The Robotic Hotwire Cutting (RHWC) approach is utilized to concrete casting in Expanded Polystyrene that has been developed to industrial scale (Feringa and Søndergaard, 2014). Currently, Odico Aps is putting forward RHWC in relation to a project design by the Danish artist Olafur Eliasson, for the Kirk Kapital HQ in Vejle [1]. Here, over 4000m2 of formwork are produced, achieving production speeds order of magnitudes faster than CNC-milling through the principal mechanics of the method (McGee, 2012). In extension of these developments, experiments at Odico are performed in abrasive wire-sawing. Through this technique, the same digital control procedures - facilitated by the internally developed control software, PyRapid – is applied to direct processing of construction materials, such as industrial marble (fig. 2, top). In further maturation of the concept, the method is being adapted in partnership development with Bäumer AG for industrial machining. Prototype production have revealed further significant reductions in machining times, in which full scale elements may be cut in matter of seconds (fig. 2, bottom).

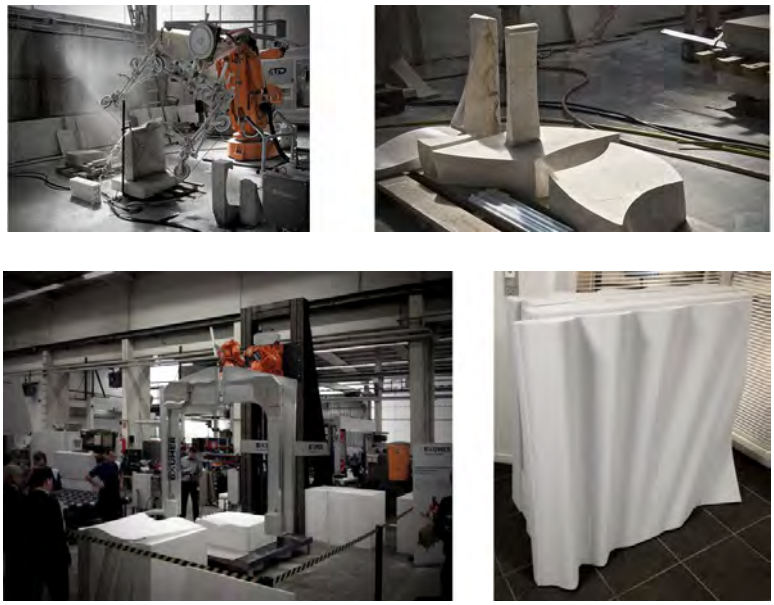


Fig 2. Robotic abrasive wire-saw cutting of marble blocks at Carrara, Italy (top, left); cut samples (top, right) Second abrasive prototype tool, developed in collaboration with Bäumer AG (bottom, right) Sample geometry cut in nonflammable acoustic foam under 16 seconds (bottom, left)

However, for a number of projects, the realization of *general* double curved structures is imperative. Here, no effective methods currently exist for architectural scale, industrial production.

In 2012, Odico Aps. tendered as part of a consortium for the realization of the before mentioned Extended Waalbridge project (fig 3). Here, the double curvature of the columns of the bridge elegantly blending with the bridge slab are dominated in a single direction. The considerable scale of the project implied large local radii (between 1- 2m) of the surfaces. Since, for this scale, CNC milling molds from EPS would have been a prohibitively ineffective method, digital manufacturing would not be economically competitive with the more traditional approach that was chosen. While developing the tender documents, Odico Aps. realized that the *Hot-Blade* cutting method presented in this paper would represent a competitive solution.


Fig. 3. The Waal Bridge Extension Impression of the artist (left) (© Zwart & Jansma)
Ongoing construction work using traditional formwork systems (right)

2 State-of-the-Art

Contemporary construction currently employs either manually produced, bespoke formwork or CNC-milling of foam molds for the realization of complex concrete structures. In addition to these techniques actuated mold systems have been explored by Danish Adapa and in the EU FP7 project TailorCrete (Jepsen et. Al., 2011; Hesse, 2012). This technique employ actuation of a flexible membrane as a casting surface; however, the method is limited to concrete prefabrication; by the casting pressure the individual systems can take; and the need for multiple casting aggregates for large volume production due the curing time for concrete elements. In addition, dynamic slip-casting for column elements is being explored (Lloret et al., 2014), as a variant of the additive manufacturing of concrete structures (Khosnevic, 2004; Lim et al., 2012). These and related methods attempt avoiding the need for formwork altogether – however does so at the cost of significant degrees of freedom, such as the capacity to realize cantilevered designs. Finally fabric formwork have been proposed and experimentally applied as an alternative technique for the casting of advanced designs (Veenendaal et al, 2011). This approach is challenged by the capacity of the fabric to achieve desired designs, as well as the unpredictability of the fabric behavior in combination with the required complexity of creating bespoke molds. A common denominator of the described developments is the requirement of shifting to entirely new modes of construction, which creates a high barrier for full scale implementation; or limits the degrees of freedom achievable compared to existing means of realization. Adversely, the here presented method proposes a production cycle which is fully compatible with current in-situ and prefabrication in concrete construction, while achieving doubly curved formwork designs at machining times more than a hundred times faster than comparable CNC-milling, the most developed and applied strategy for industrial scale production.

Double curved surfaces with positive Gaussian curvature can in a vast majority of cases be described via swept splines. The term “splines” nowadays refers to piecewise polynomial or rational functions used in CAD systems to model curves and surfaces. However, prior to the introduction of computers, which began in the 1950s, the word was used for thin wooden rods the shapes of which were manipulated by the placement of so-called “ducks” at various points to create a naturally smooth curve for drawing designs. These were used in ship building and, later, in the aviation and automotive industries. The placement of the ducks simulates the placement of ribs in the

hull of the ship, and hence the curve drawn by following the spline is an accurate reflection of the natural shape adopted by the planks forming the ship's hull. The use of splines for the storage and transmission of a design goes back to the Romans, in the form of *physical* templates for the ribs of ships (Farin, et al., 2002). Splines and ducks suitable for *drawings* of ship designs were developed later, perhaps in Hull in the 1600s.

Fig. 4. Design of the Concorde wing-section using physical splines (1964) (© Bristol Archives)

The mathematical *shape* of a physical spline can be described exactly, although it requires the use of so-called *elliptic functions*, which are nonlinear in nature. The correct mathematical model for an elastic rod bent by a force at one end with the other end fixed was given by James Bernoulli in 1691 (Truesdell, 1983). In his approximation of the solution for the case that the ends of the rod are at right angles to each other, he recognized that the solutions would require non-standard functions. Later, in 1743, Bernoulli's nephew, Daniel, suggested the problem to Euler, who then, in an appendix to his famous treatise on the calculus of variations found all possible shapes for these so-called *Euler elastica* (Euler, 1744; Love, 1906).

Geometry Rationalization

The presented geometry rationalization approximates the physical behavior of the the Hot-Blade in order to convert arbitrary input surfaces to producible geometry. The HotBlade is fixed between two robot arms, which enable us to choose the location and rotation of the blade's ends. The shape of the blade is the curve that, subject to the endpoint constraints, minimizes the elastic energy. These curves are the above-mentioned Euler elastica or elastic curves. Before discussing the approximation of a CAD surface, let us consider the class of surfaces defined by this cutting process, namely the surfaces swept out by continuously varying families of planar Euler elastica. A planar curve is geometrically determined by its curvature function $\kappa(s) = \theta'(s)$, where \theta is the angle function of the unit tangent.
One can show that the equation defining an elastica is the normalized pendulum equation $\theta''(s) = -\sin(\theta(s))$ and the solution is the curve:

$$C_k(s) = \Big(2E(s,k) - s, 2k\big(1 - \mathrm{cn}(s,k)\big)\Big), \qquad k = \theta'(0)/2,$$

where cn(*s*,*k*) and *E*(*s*,*k*) are standard elliptic functions depending on a parameter *k*.

Applying all possible dilations, translations and rotations to C_k, one obtains all possible elastic curve segments. Allowing all of these parameters to vary with time, and then generating the time sweep so defined, one obtains all possible elastica-swept surface patches. One can implement this numerically, to obtain examples (Fig.5).

Fig. 5. Elastica surfaces generated through implementation of the above formulation in Matlab

When rationalizing a CAD surface to Euler-elastica for Hot-Blade cutting, the surface is segmented into patches that can be approximated by surfaces of the type exemplified in Fig. 5. We essentially do this simply by finding planar curves on the original surface and then approximating these by segments of planar elastic curves.

2.1 Curve Approximation

Given a parameterized planar curve segment we wish to find a piece of an elastic curve which has the same shape. We do this via an optimization algorithm that minimizes the distance between two curves. By choosing a standard parameterization, we are able to describe any elastic curve segments by four control parameters, which determine the length and shape of the segment. Three more parameters determine the position and rotation of the curve in the plane. The distance between the given curve and any elastic curve is thus a function of the seven control parameters.

The approximation algorithm has two steps: first, we analyze the geometry of the given curve in order to find control parameters for an elastic curve segment, which has the same overall shape (Fig. 6). Then, starting from this initial guess, we tweak the parameters, using the optimization tool IPOPT (Wächter and Biegler, 2006), until we get the closest fit .We can do this either with or without endpoints fixed.

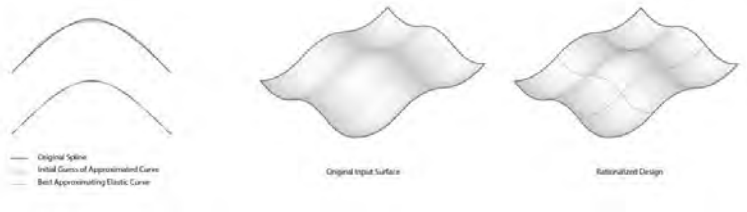


Fig. 6. Original spline curve (blue) and initial guess for approximating elastic curve (red dotted) & Original spline curve (blue) and best approximating elastic curve (green) (left). Input NURBS surface (center); Rationalized surface (right)

2.2 Surface Approximation

We now consider a given CAD surface, and we want to approximate it by a surface that can be obtained by moving elastic curves through space. From the CAD design we extract planar curves on the surface and approximate each of these by an elastic curve. By interpolating the control parameters we obtain a rationalized design - a new surface, which is swept out by elastic curves moving through space. For larger designs we need to segment the surface into pieces that can be cut individually. Because we control the endpoints and directions of the blade, we can ensure smooth transition from one piece to another (Fig. 7).

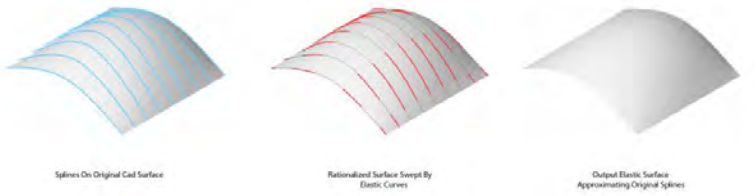


Fig. 7. A selection of planar curves (splines) on the original CAD surface (left); The original surface with elastic curves that approximate the splines. Note that these curves do not lie exactly on the surface (center); Rationalized surface swept by elastic curves (right)

3 Surface Segmentation

A number of segmentation procedures are developed, targeting three production constraints: a) plain segmentation when exceeding the dimensions of the input EPS work object; b) instability of the blade due to multiple inflection points, or c) cutting the same area multiple times due to rotation of the blade profile. Fig. 8, left illustrates an example of a surface with too many inflection points. An inflection point is a point where the sign of the curvature changes; in other words the tangent at the point will cut the curve in two. We use a subdivision scheme to find the inflections. Analysis of

one of the curves shows six inflection points and since many inflection points on the curve make the blade less stable, segmentation is required.

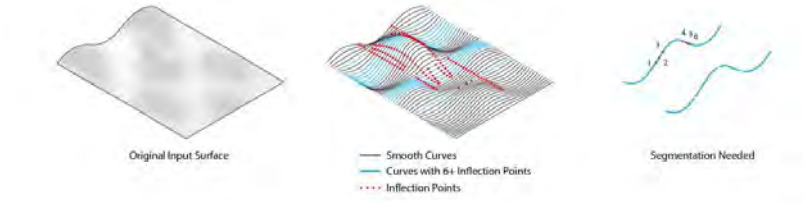


Fig. 8. An input surface (left); The Hot-Blade planar cuts with inflection points (center); One of the cuts close up (right)

Assuming the rationalization of each cut is curvature continuous, there will be the same number of inflection points on the cut and the rationalization. Two exceptions to this are inflection points near the edge of the cut that may disappear, and pairwise inflections close to each other, which may cancel out, just like pushing out a small dent.

Taking the above into account, we propose the following algorithm.

1. Find the planar curves on the surface.
2. Calculate inflection points for each curve.
3. Segment the surface into a grid of blocks.
4. For each block test if there are more than two inflection points; if so try to
   a. Move the block if there are overlaps to improve.
   b. Remove inflection points close to each other.

   If there still too many inflection points continue to step 5.
5. Take two new blocks, each of the same size as the original block, and place them so that they overlap both each other and the two adjacent blocks in the row. Go to step 4.

In this algorithm we can control whether we keep the same number of blocks in all rows or not. This affects the aesthetics of the segmentation. In the overlap of the blocks we choose a cutting plane such that the segmentation follows the geometry. An example of the output of this algorithm can be seen in Fig. 8 (right), showing the surface subdivision.

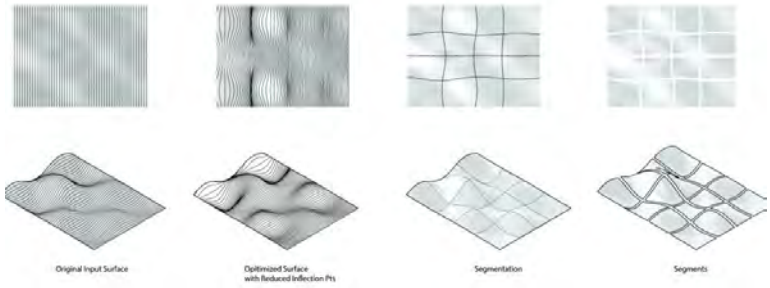


Fig. 9. Segmentation schemes

The problem of cutting the same area multiple times arises when rotation of the blade in the cutting direction is allowed (Fig. 9, column 2 from the left). We see here that the curves intersect each other, and thus part of the surface will be cut multiple times, which is undesirable. In most cases this problem can be solved by segmenting the surface, as described above. We only need to add a test for intersecting curves in step 4.

4 Dataflow and Robotic System Configuration

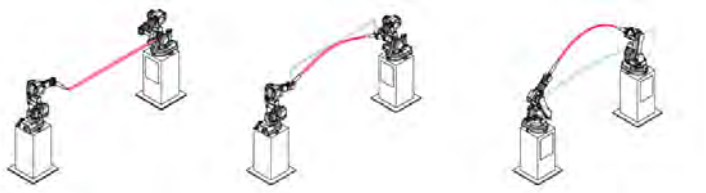


Fig. 10. Deformation of the blade through orientation and positioning of the two end-effectors

The experimental setup consists of three robots. Robot 1 holds the EPS work object, which is to be cut, and moves the block linearly through space, thus acting in principle like a conveyer belt. Robots 2 and 3 control the ends of the HotBlade thereby determining its shape and its position in relation to the EPS block. When the geometry rationalization is completed, we know a set of planar elastic curves on the rationalized surface. The curve segments which lie on the surface are shorter than the HotBlade cutting tool, but since we know not just the curve segments, but the entire curves we can easily extend the curves to the required length, i.e. the length of the HotBlade. These extended curves are the target shapes for the HotBlade during the cutting. We extract the relevant data for the extended curves, that is, we find the coordinates for the endpoints and the tangents at the endpoints. The endpoint coordinates determine

the position of the tools of robots 2 and 3 relative to the EPS block. The tangents determine the rotation of the tools, which in turn controls the shape of the blade.

For our experiments the robots were given 51 targets. That is, for each block that was to be cut, we provided 51 sets of positions and rotations for the tools of robots 2 and 3. The robot program then interpolates between these targets to follow a smooth path from the first to the last target, thus moving the blade while changing its shape, resulting in an EPS surface of the rationalized design.

5 Blade Mechanics and Cutting Experiments

The main cutting tool used in the process is a thin metal strip - usually referred to as a blade - made of a nickel-chromium super alloy. The blade is pre-heated to a temperature of 300-400°C by means of Joule heating and then it is slowly brought into contact with an EPS block to produce melting, and subsequently to form or cut the block into a desired shape (also referred to as thermal cutting). At such high operating temperatures, the blade has to be displaced (or bent) into an elastic shape with predefined curvature and at the same time maintain its elastic and flexibility properties. Using FEM simulations, the effect of mechanical properties on the target geometry was investigated and a particular material was chosen to ensure smooth cutting. The blade is attached to two robots, one at each end, by specially designed sandwich based holders to ensure strong and safe supports during all cutting operations. The physical displacement of the blade is achieved by moving the robots into an appropriate position, at the same time maintaining the elastica-strain-curvature relations. The temperature dependent variations of the blade shape are to be incorporated in the computational algorithm to secure proper shape representation (Fig 11).

Fig. 11. tri-robot hot-blade cutting configuration.

Two experiments were designed and performed in order to test the utility of the setup. In the first experiment a convex doubly curved surface was cut. The curvature of the blade was continually changing during cutting in order to test the limit of complexity that can be achieved and ensure proper geometrical representation. The presence of two inflection points on the discretized surface was considered as a possible problem, but the experiments showed that it does not make the blade unstable, since the robots compensate with the angles of the holders and the curvatures involved were moderate. Good surface quality was achieved at cutting with an absolute speed of motion of 7mm/s. The EPS block to be cut had the dimensions of 600x600x600mm.

The second test aimed to cut a number of EPS blocks and then assemble them into a single structure that should represent a ready-made mold for concrete casting. Different discretized pieces of doubly curved surfaces of both convex and concave types, as well as hyperbolic surfaces (negative Gauss curvature), were successfully cut with the setup. The size of each individual block was approx. 600x785x600mm, resulting in an assembly of size 1800x2345mm, comparable to the size of production frame molds. The cutting experiments are currently continued for production of doubly curved concrete panels with expected completion December 2015.

6 Formwork Systems and Production Workflow

The efforts described in the previous chapters outline the general method for the cost-effective production of doubly curved formwork in Expanded Polystyrene. From this, the following process is developed (Fig. 12):

Fig. 2. General production workflow diagram: segmentation (left); cut foam (center) and in-situ mould (right)

The cyclical workflow links conventional CAD-modelling operations with the robotic Hot-Blade fabrication and standard concrete casting techniques. This requires the rationalization and segmentation of geometry types before rebuilding the geometry to the constraints of the blade, robot work envelope, work object dimensions and tolerances. After the input geometry has been translated to segments of swept Euler elastica surfaces and data deducted for tri-robot motion, EPS-mold pieces are produced. The mold pieces are subsequently used in combination with existing pre-fabrication and in-situ workflows. For element pre-fabrication, molds are mounted on vibration tables and sides enclosed with metal or wooden frames. For in-situ applications, mold pieces are used in combination with standard scaffolding modules for casting pressure support. These applications ensure a full compatibility of the end-products of the Hot-Blade with established industry workflows, critically ensuring a low barrier to adoption.

7 Conclusion

A general purpose robotic fabrication method for producing doubly curved formwork has been presented. The efficacy of the method has been demonstrated through geom-

etry rationalization and pilot production of a sample formwork panel design. The method is being implemented for industrial scale fabrication by one of partners of the research consortium, and the identified challenges are being addressed through this work.

Acknowledgements The work presented in this paper is part of the larger 3-year research effort, 'BladeRunner' established and generously supported under the program of the Innovation Fund Denmark for advanced technology projects. The project is conducted by the partners Odico Aps (project lead), the Technical University of Denmark, Department of Applied Mathematics and Computer Science and Department of Mechanical Engineering, the Danish Institute of Technology; GXN A/S and Confac A/S.

References

Euler, L 1744, Methodus inveniendi lineas curvas maximi minimive proprietate gaudentes; Additamentum I: de curvis elasticis.

Farin, G 2002, 'A History of Curves and Surfaces in CAGD', in Handbook of Computer Aided Geometric Design, North-Holland Publishers, Amsterdam, pp. 1-23.

Feringa, J and Søndergaard A 2014, 'Fabricating Architectural Volume' in Kohler, M and Gramazio, F (eds), Fabricate : negotiating design and making, gta-Verlag, Zürich, pp. 44-51.

Hesse, P 2012, 'TailorCrete, Flight Assembled Architecture' in Architekturteilchen. Modulares Bauen im Digitalen Zeitalter, Köln, pp. 126-127, 164-165.

Jepsen, C, Kristensen M, Kirkegaard, P 2011, 'Dynamic Double Curvature Mould System' in Gengnagel, C, Kilian, A, Palz, N and Scheurer, F (eds), Computational Design Modeling : Proceedings of the Design Modeling Symposium, Springer, Berlin, pp. 291-300.

Khosnehvis B, 2004, 'Automated Construction By Contour Crafting – Related Robotic and Information Technologies', in Journal of Automation in Construction Special Issue: The Best of ISARC 2002, vol.13, no.1, January 2004, pp 5-19.

Lim, S, Buswell, RA, Le, TT, Austin, SA, Gibb, AGF and Thorpe, A 2012, 'Development in Construction-Scale Additive Manufacturing Processes', Automation in Construction, vol. 21, no. 1, pp. 262-268.

Lloret, E, Amir, R, Shahab, Mettler, L, Flatt, RJ, Gramazio, F, Kohler, M and Langenberg, S 2014, 'Complex Concrete Structures: Merging Existing Casting Techniques with Digital Fabrication', Computer-Aided Design, Elsevier, Amsterdam, NL, vol. 60, March, pp. 40–49.

Love, A 1906, A Treatise on the Mathematical Theory of Elasticity, Cambridge University Press, Cambridge, UK.

McGee, W., Feringa J., Søndergaard, A Processes for an Architecture of Volume: robotic hotwire cutting. In: Brell, S., Braumann, J (eds) Robarch 2012: Robotic Fabrication in Architecture, Art & Design. Springer Verlag, Vienna, pp 62-71.

Truesdell, C 1983, 'The Influence of Elasticity on Analysis: the Classic Heritage' Bulletin of the American Mathematical Society, vol. 9, no. 3, pp. 293-310.

Veenendaal, D, West, M and Block, P 2011, 'History and Overview of Fabric Formwork: using Fabrics for Concrete Casting', Structural Concrete, Ernst & Sohn, Berlin, vol.12, no 3.

Wächter,A., and Biegler L.T, 2006, 'On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming.' Mathematical Programming, Springer Berlin, 106(1):25-57.

[1] http://www.domusweb.it/en/news/2011/12/01/kirk-kapital-a-s-by-eliasson.html

Designing for Robotic Hot-Blade Cutting

David Brander[1] Andreas Bærentzen[2] Kenn Clausen[3] Ann-Sofie Fisker[4] Jens Gravesen[5] Morten N. Lund[6] Toke B. Nørbjerg[7] Kasper Steenstrup[8] Asbjørn Søndergaard[9,*],

[1] Technical University of Denmark, Dept. of Applied Math. and Comp. Sci.
dbra@dtu.dk

[2] Technical University of Denmark, Dept. of Applied Math. and Comp. Sci.
janba@dtu.dk

[3] GXN A/S
kec@3xn.dk

[4] Technical University of Denmark, Dept. of Applied Math. and Comp. Sci.
ansofi@dtu.dk

[5] Technical University of Denmark, Dept. of Applied Math. and Comp. Sci.
jgra@dtu.dk

[6] GXN A/S
mnl@3xn.dk

[7] Technical University of Denmark, Dept. of Applied Math. and Comp. Sci.
tono@dtu.dk

[8] Technical University of Denmark, Dept. of Applied Math. and Comp. Sci.
khor@dtu.dk

[9] Odico Formwork Robotics Aps.
asbjorn@odico.dk

Abstract. This paper presents a novel method for generation of doubly curved, architectural design surfaces using swept Euler elastica and cubic splines. The method enables a direct design to production workflow with robotic hot-blade cutting, a novel robotic fabrication method under development by authors of the paper, which facilitate high-speed production of doubly curved foam molds. Complementary to design rationalization, in which arbitrary surfaces are translated to hot-blade-cuttable geometries, the presented method enables architects and designers to design directly with the non-trivial constraints of blade-cutting in a bottom-up fashion, enabling an exploration of the unique architectural potential of this fabrication approach. The method is implemented as prototype design tools in MatLAB, C++, GhPython and Python and demonstrated through cutting of expanded polystyrene foam design examples.

Keywords: robotic fabrication, Hot-Blade, digital design, EPS-molds, cost-efficiency, concrete structures

1 Introduction

In contemporary architectural practice, a rising number of projects employ advanced building geometries, which departs with the orthogonality of mainstream construction, incorporating digital design tools and manufacturing for the realization of expressive or dynamic design features [Pottman 2007]. A group of projects within this category, such as Kagamigahara Crematorium (Toyo Ito Architects, 201) and Waalbridge Extension (Zwart & Jansma, 2015), rely on the doubly curved geometries which may be constructed either via production of manual formwork, which

relies on digitally produced guides to bend plate material in place over large radii. Alternatively, large scale CNC-milling of either foam molds for concrete casting or direct milling of construction materials are employed enabling the realization of shorter radii designs with more detail and surface controls. Such projects include for example Spencer Dock Bridge (Amanda Levete Architects, 2010), Louisiana State Museum and Sports Hall of Fame (Trahan Architects, 2013), Museum Foundation Louis Vuitton by Gehry & Associates (Paris, 2014); the Nordpark cable railway by Zaha Hadid Architects (Nordpark, 2007), the Metz Pompidou by Shigeru Ban (Metz, 2010).

Fig. 1. Louisiana State Museum and Sports Hall of Fame, courtesy Trahan Architects (left). Kagamigahara Crematorium, Courtesy Toyo Ito Architects (right)

However, none of these general construction processes provides a cost-effective option for general construction, and projects of this type therefore require extraordinary budget frameworks for realization: manual onsite formwork processing in this category is a highly laborious and demanding process, with resulting difficulties in cost-engineering to follow; large scale CNC-milling on the other hand, provides cost transparency due to the digital nature of the process – however the mechanical principle of CNC-milling, which subtracts material through incremental removal, is inherently slow when applied to architectural scale production, hence resulting in exuberant machining times and high costs.

Recent developments in architectural robotics and digital manufacturing have seen the emergence of a number of approaches to diversify the machining options available, with the purpose of realizing structures of more advanced geometries. This includes actuation of a flexible membrane as a casting surface [Jepsen et. al, 2011; Hesse, 2012]; dynamic slip-casting for column elements [Lloret et. Al 2014], as a variant of the additive manufacturing of concrete structures [Khosnevic 1998, Lim et al 2012]; fabric formwork applied as an alternative technique for the casting of advanced designs [Veenendaal et. al 2011]; spatial wire cutting [Rust et al 2015], as well as large scale robotic hot-wire cutting of EPS molds by authors of this paper [reference omitted for blinding purposes, 2014]. None of these approaches however, are capable of delivering combined 1) unconstrained degrees of freedom which enables general purpose realization equivalent to that of CNC-milling; 2) machining efficiencies which significantly supersedes that of CNC-milling; 3) process predictability which ensures the delivery of a pre-controlled geometry.

2 Robotic Hot-blade Cutting

In an effort to develop a new manufacturing process, which would satisfy these criteria, authors of this paper initiated in 2012 the Bladerunner project, which targets the cost-effective production of double curved foam molds. The technique developed in this effort - dubbed Robotic Hot-blade cutting - employs a multi-robotic process, in which a 18-axis cell consisting of 3 industrial manipulators translates a flexible, heated blade through expanded polystyrene blocks in a thermal cutting process, while controlling the distance and rotation of end-effectors to achieve variable cross-section curves along the trajectory of cutting sequences [reference omitted for blinding purposes, 2016]. Pilot production experiments currently under development seek to explore and demonstrate the applicability of this method for production of pre-fabricated concrete elements under a general CAM paradigm, in which arbitrary design input – understood here as geometry which is conceived without particular regard to the specific constraints of the process – is rationalized for hot-blade production using a set of algorithms developed within the project [reference omitted for blinding purposes, 2016]. These early developments point to the perspective of a highly time-efficient production method, up to 126 times faster than comparable CNC. However, complementary to a top-down process of rationalization, a second trajectory is also possible, in which the geometric constraints of the hot-blade cutting is incorporated already in the design process, thus operating under a generative design paradigm. The work in the following chapters outlines tools and processes that can facilitate such approach.

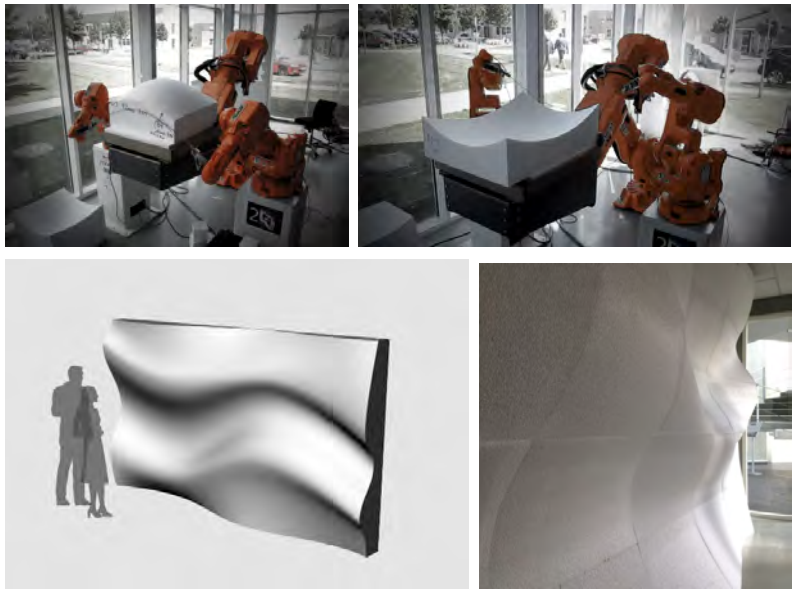


Fig. 2. Bladecutting experiments in progress. Top: 18-axis tri-robot hot-blade pilot-cell. Below: concrete panel design and cut foam result.

3 Designing with elastica

An Euler elastica is the shape assumed by an elastic rod with planar constraints of position and tangents placed only on its endpoints. A planar curve is geometrically determined by an angle function $\theta(t)$, the angle between the tangent and some fixed direction. The angle function for the elastica are given by solutions of the normalized pendulum equation $\theta'' = -\sin\theta$, a nonlinear equation the solutions of which can fortunately be given explicitly in terms of elliptic functions.

Mathematically, the correct model for an elastica was given by James Bernoulli in 1691 (see Truesdel, 1983). He approximated the solution for the case that the ends of the rod are perpendicular to each other, recognizing that non-standard functions were needed for an analytic expression. The problem was subsequently suggested to Leonhard Euler in 1743, who gave all possible shapes for the elastica in his famous treatise on the calculus of variations (Euler (1744)).

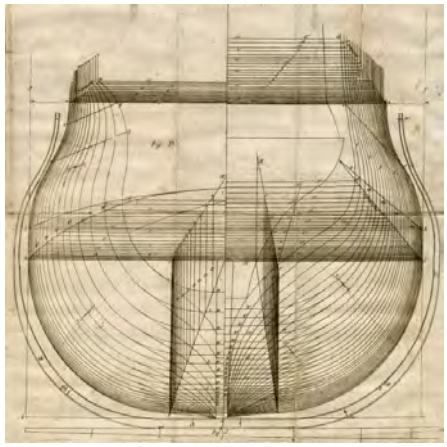


Fig 3. Use of physical splines for ship-hull manufacturing. William Sutherland, The Shipbuilders Assistant: or, Some Essays Towards Completing the Art of Marine Architecture (London, 1711)

People have in fact been designing with elastica for centuries, albeit in a physical rather than mathematical format. Prior to the introduction of computers for draughting in the shipping, aviation and automobile industries, which began in the 1950s, the curves needed in the designs were created by tracing the shapes of thin wooden rods, known as splines, manipulated by the placement of so-called “ducks” at various points to create a naturally smooth curve. This practice started in the ship-building industry, where the placement of the ducks simulates the placement of ribs in the hull of the ship: hence the curve drawn by following the spline is an accurate reflection of the natural shape adopted by the planks forming the ship’s hull. The drawing took place at the loft of the shipyard, hence the word “lofting”, now used in the CAD industry. Going further back in time, splines were used for the storage and transmission of designs in Ancient Rome, in the form of *physical* templates for the ribs of ships (see Farin, et. Al 2002).

When computers became cheaper and more powerful, a desire for electronic storage and editing appeared. The word “spline” now began to be used for piecewise polynomial or rational curves used in design. Paul de Casteljau at Citroën and Pierre Bézier at Renault used what are now known as Bézier curves to describe the designs. In the USA, Carl de Boor at GM used B-splines (basis splines) for the designs. In the aircraft industry, at Boeing, similar developments took place.

3.1 Design vs. rationalization for hot-blade cutting

For a CAD surface to be produced using hot-blade cutting, it needs to be segmented into suitable pieces and each surface segment then swept by planar curves that are subsequently approximated by elastic curve segments. We have described this rationalization process in recent and forthcoming work.

An alternative to rationalization of a CAD design is to provide design tools that allow designers to create fabrication-ready surfaces. There are a number of reasons for doing this: firstly, the rationalization of an arbitrary design is non-trivial and in general can result in some regions of the surface needing to be produced by another method such as milling. Secondly, a design tool can give the designer control over the cast-lines between the segments, which will in many cases be visible from close range.

A third reason is the additional complexity arising when we consider a surface created by more than one cut to the same EPS block. For example, consider the surface shown on the left in Figure 4. By cutting the same EPS block twice, the second time with a 90 degree rotation, the surface on the left in Figure 5 is produced. Now the surface on right in Figure 4 approximates the first surface very well at the end-points of the cutting blade, but deviates slightly in the middle. Such an approximation is likely to arise in surface rationalization, because we will usually need the patches to fit together with tangent continuity, which requires a little more freedom away from the patch edges. Doing the same two cuts with the new surface results in the surface on the right in Figure 5, and here we can see that the intersection curves are no longer the same as the design.

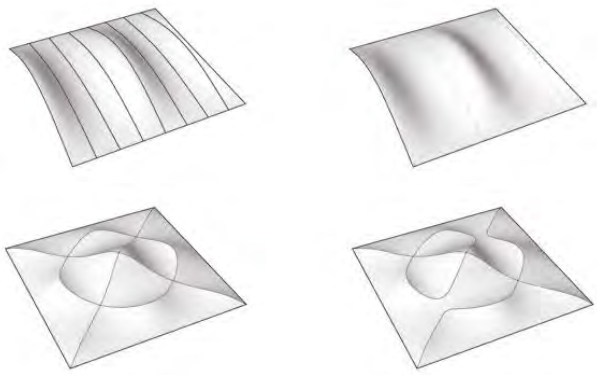


Fig. 4 & 5. (Top) Matlab generated surface. (Bottom) rotated double cut of the same design

Of course there are solutions for this kind of issue in the rationalization approach, but this scenario illustrates the kind of advantages one has with a fabrication-ready design tool.

3.2 Single block designs

For the simple case of just one block, we design with curves of the desired length, i.e. the length of the cutting tool. During cutting the cutting tool is kept in a horizontal plane perpendicular to the cutting direction. The data needed for the robot movement is thus simply the positions of the ends of the blade and the angles of its ends relative to horizontal. The positions are given as *y*,*z*-coordinates (the *x*-coordinate describes how far the curve is in the cutting direction), see Figure 6. In the design space (e.g. Rhino) we know the position of the design curves relative to the EPS block, so we can obtain the robot data directly from the design curves by computing the positions of the endpoints relative to the EPS block, and their angles relative to horizontal.

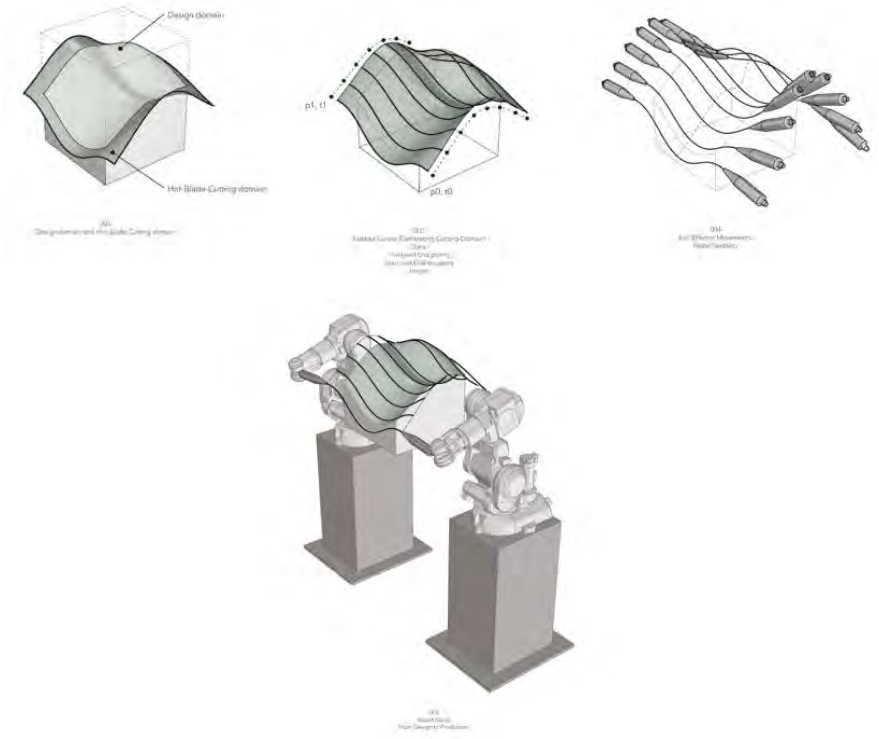


Fig 6. Configuration of input data

The plugin for the discrete elastica ensures that we get a representation of the final design in Rhino, before going to production. In the following, we describe the numerical algorithm used to find this solution given end points, tangents at the end point, and the length of the curve (see Bruckstein et al (2001)). This method does not find the elastica in terms of the elliptic functions: instead we return to the defining property of elastic curves, namely that they minimize $\int \kappa^2 ds$. In a discrete setting where we represent the curve using n line segments of equal length, the analogous energy for the piecewise linear curve is:

$$F_a(\alpha) = \sum_{i=0}^{n-1} \alpha_i^2$$

where α_i is the turning angle at segment i as illustrated in figure 7.

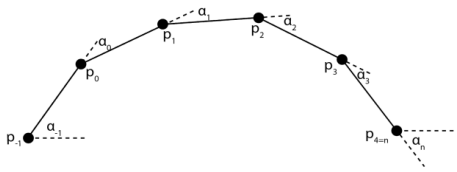


Fig 7. Turning angles

Thus, to find the discrete curve, we minimize F_a subject to two constraints that both serve to enforce the boundary conditions. To ensure that the tangent at the last point has the correct direction, we require that

$$0 = F_t(\alpha) = \sum_{i=0}^{n-1} \alpha_i - (\alpha_n - \alpha_{-1})$$

where α_{-1} and α_n are the angles that correspond to the direction of the first and last line segment respectively. Clearly, we also require that the curve ends at the right point. This is taken care of by the second energy

$$F_{p(\alpha)} = \left\| \sum_{j=-1}^{n-1} \left[\cos\left(\sum_{k=-1}^{j} \alpha\_k \right), \sin\left(\sum_{k=-1}^{j} \alpha_k \right) \right]^{\mathrm{T}} - (p_n - p_{-1}) \right\|$$

We can construe the discrete elastic curve problem as an inverse kinematics problem. In this light, F_p simply ensures that the end of the curve (end effector) coincides with p_n - the end point of the elastic curve.

Fig 8. Turning angles for a discrete elastica

In practice, we find the elastic curves using a two step procedure. In an outer loop, we reduce F_a and in the inner loop, we solve for $F_p = 0$ and $F_t = 0$. F_a and F_p are minimized using gradient descent whereas F_t can be solved exactly in each step. The outer loop runs for a fixed number of iterations whereas the inner loop terminates when a desired tolerance has been reached. Fig. 11. shows a test. The green figure-eight is an analytic elastic curve. The black curve on top is the discrete approximation computed from the boundary conditions shown in red.

3.3 Examples

The images below show a surface defined as the graph of a bivariate function

$$z = f(x,y) = k\cos\left(2\left(\frac{\pi}{2} - y\right)^4\right)\sin(x)\exp(-0.1x^2) \quad [x,y] \in \left[-\frac{\pi}{2},\frac{\pi}{2}\right] \times [-0.75,0.75]$$

z = f(x,y) =k(2(/2-y)4)(x)(-0.1 x2) [x,y] [-/2, /2][-0.75,0.75]

As the initial design created using a Python script in Rhino, as a rationalized surface and as a styrofoam block cut with the hot blade.

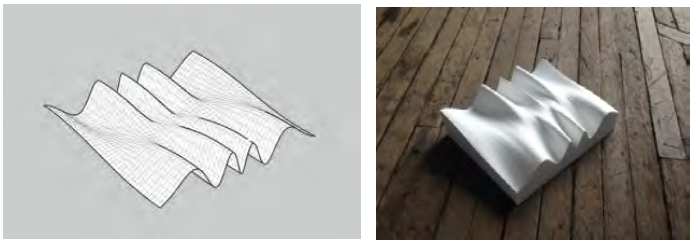


Fig 9. Examples of a generated and a cut surface.

3.4 Design using multiple blocks

For the more complex case with several blocks a more advanced procedure is used to ensure a smooth transition from one block to the next. Consider a curve design that passed over two blocks (see figure 10, left). We need to produce this in two cuts – one per block – and we want the two block segments to match at the boundary after cutting.

If we run our plugin independently on the two parts of the curve, we would automatically obtain the smooth transition, but we would be unable to get the necessary data for the robots, since the curves (which are inside the blocks) are shorter than the cutting tool and we have no quick way to extend these while preserving the elastic properties. If we simply extend the original curve and then use our plugin on parts with the desired length, we do not ensure smooth transition (see Figure 10, right).

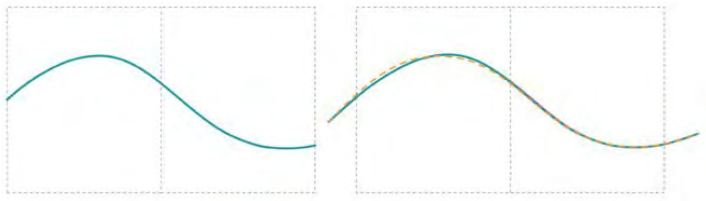


Fig 10. In blue an exact elastic curve and in black a discrete approximation calculated from the boundary conditions shown in yellow.

The solution is, instead of finding a discrete representation of the elastic curve that models the shape of the cutting tool, we find an analytic mathematical description of the elastic curve in terms of elliptic functions. This requires a more cumbersome optimization in order to find the parameters that describe the rationalized curve. However, when these parameters are found, the entire (infinitely long) elastic curve that contains the required segment is known. It is then simple to extend the segment to an elastic segment of the length of the blade (see Figure 11) and from this extract the position data for the robots.

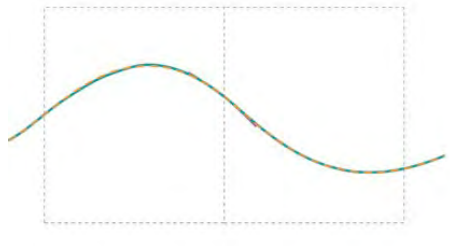


Fig 11. Transitions between elastica

4 An alternative approach: Bézier curves as a proxy for elastic

Historically the use of cubic splines as a design tool was often motivated by saying that they are a good substitute for real physical splines. This is justified by the fact that if the speed of a curve is constant then the square of the curvature is the same as the square of the second derivative and if the latter is minimized we obtain exactly a cubic spline (see Yamaguchi (1988)). Now a cubic curve does not have constant speed unless it is a straight line; but if the control polygon of a cubic Bézier curve is reasonably well behaved then the curve is close to an elastica: see Figure 12, where 24 Bézier curves are plotted together with elastic curves having the same endpoints, end tangents, and length.

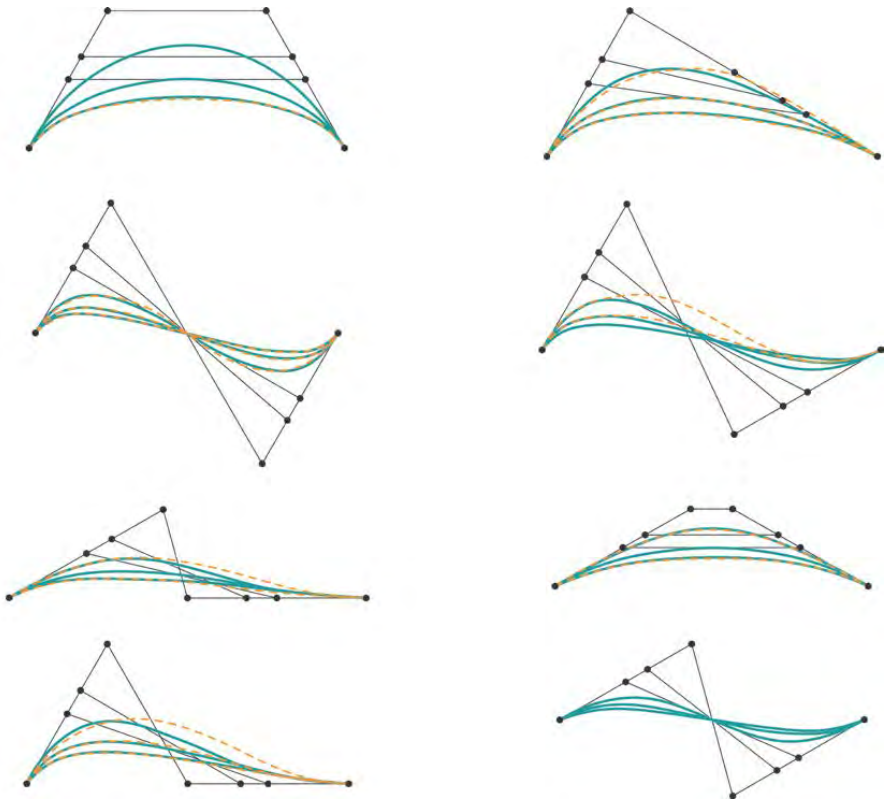


Figure 12: In blue Bézier curves, in black their control polygons, and in dashed red elastic curves with the same endpoints, end tangents, and lengths.

If the angles in the control polygon are not too acute then there is very little difference between the Bézier curve and the elastic curve of the same length and end conditions.

Based on this observation we have implemented a design tool in Rhino™ where the surfaces and their rationalization are very close. The idea is that we imagine space filled with EPS-blocks and define our surfaces such that the parameter curves have exactly one planar cubic piece in each block. As a simple example consider Figure 13, where we have three blocks and have defined three Bézier curves at both the front and the back of the blocks in such a way that they have common endpoints and tangents, i.e., they form two tangent continuous curves (see Figure 14). The surface is defined by a lofting process. We can of course have several layers of blocks and we could also have included more curves in the middle of the blocks. The inputs are planar curves with exactly one cubic piece in each block, and the surface is defined by lofting.

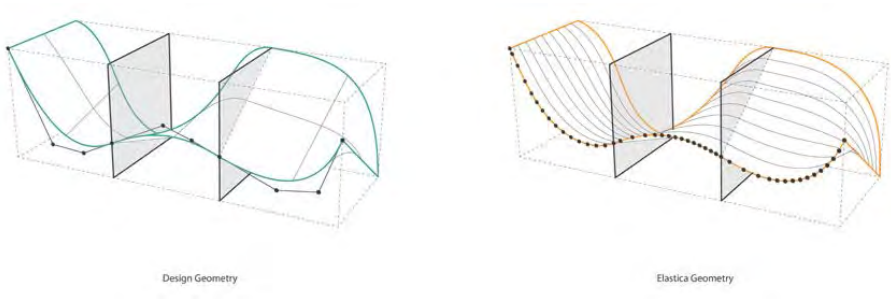


Figure 13: A Rhino design tool. Top: three blocks with two tangent continuous curves. Lower left: lofted Bézier curve surface. Lower right: rationalized elastica curve surface.

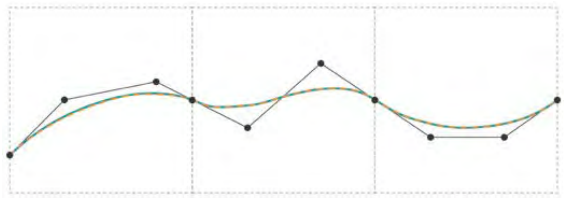


Figure 14: The tangent continuous construction. In each block we have a cubic Bézier curve and we require that adjacent curves have control polygons the first and last legs of which form a single line segment. In dashed red we have plotted the true elastica having the same length and boundary conditions as the Bézier curves.

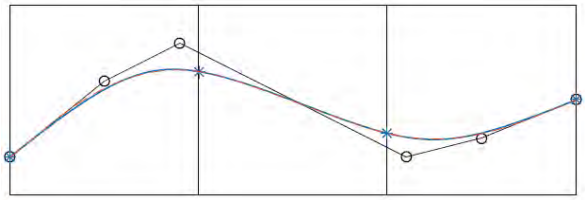


Figure 15 : The curvature continuous construction. A single cubic spline curve where the images of the knots are on the block boundaries. In dashed red we have plotted the true elastic curves that have the same length and endpoints as the polynomial pieces. The tangents corresponding to the extreme points of the cubic curve are also prescribed.

One can achieve a curvature continuous construction by replacing the sequence of Bézier curves by a single planar cubic spline (with simple interior knots). Between the knots we have a cubic polynomial, so we if we require that the image of the knots are on the block boundaries then we obtain a single cubic polynomial piece in each block (Figure 15).

If we replace the spline with an elastica having the same length in each block as the spline, passing through the images of the knots, and having the same tangent angles in the beginning and end as the spline curve, we obtain almost the same result. This corresponds exactly to the classical design using a physical spline and ducks. We just have to place the ducks at the images of the knots.

5 GH Workflow

For the development of design experiments as well as participatory workshop design sessions, a toolset is developed in McNeel Grasshopper, implementing the above approaches. The toolset is linking the full cycle of research, innovation, implementation and production, creating a framework for geometric operations consistent with the robotic setup. The overall logic of the workflow connects conventional digital modeling approaches with the robotic hot-blade process. This requires the identification and rationalization of geometry types before rebuilding the geometry to the accuracy of the robot, EPS-segmentation and tolerances.

The Grasshopper-definition is developed with the purpose of designing with rationalization through Euler elastica. It is a Real-Time process that allows for fast interpolations from design to production and ensures a smooth curve continuity transition from one block to the next. The tool is very flexible and allows for large variation of form typologies when designing with single or multiple cuts in the design explorations.

The setup is part of a larger digitized workflow; *"Interpolation of Geometry"*, *"Euler Elastica Approximation"*, *"Mathematical Elastica Extension"*, which is is a linear process that allows for feedback loops when data or geometry are outside of preset conditions and/or needs changes. The Grasshopper tool "*Design with Elastica*" inputs arbitrary surfaces and/or curves, converts them into planar elastica curves that describe the cut-direction and the movement of the robot-setup. The setup is structured in four overall processes; *"Global Parameters"*, *"Input"*, *"Process (Machine)"*, *"Output (Export)"*.

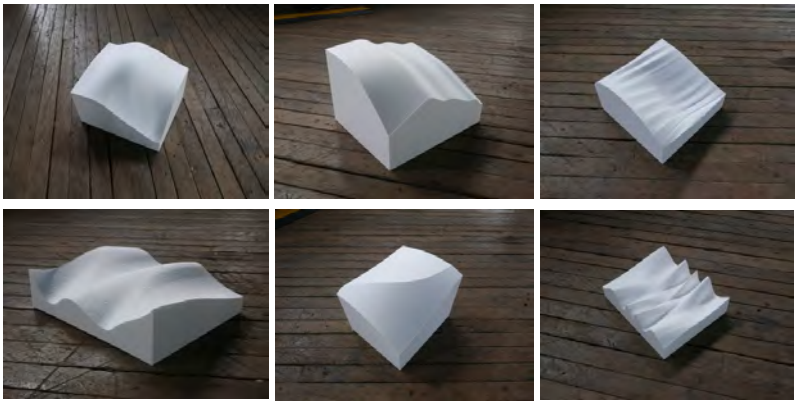
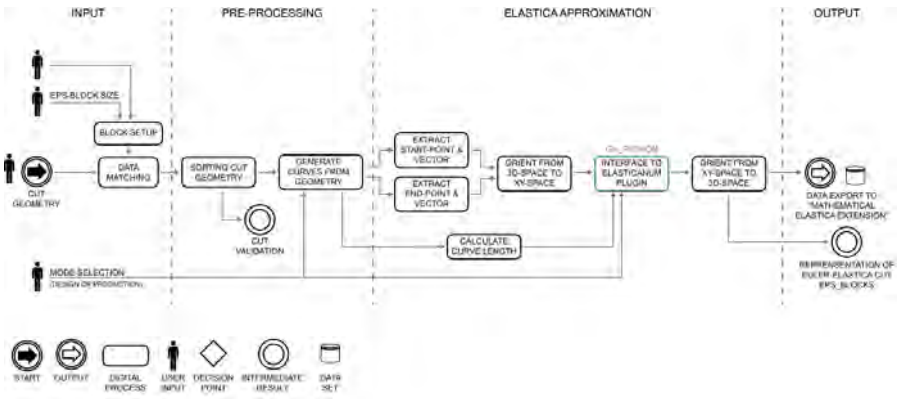


Figure 16: pre-test cuts from the workshop. From the left: while single, continuously swept surfaces are readily achievable through rationalization, the ripple and curvature effect on the right

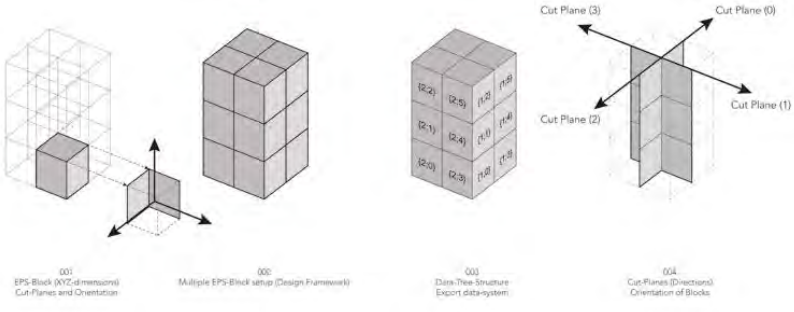
most samples requires careful alignment with the blade directions, and hence is difficult to obtain aside from directly controlling it in an elastica-swept surface.



5.1 Global parameters

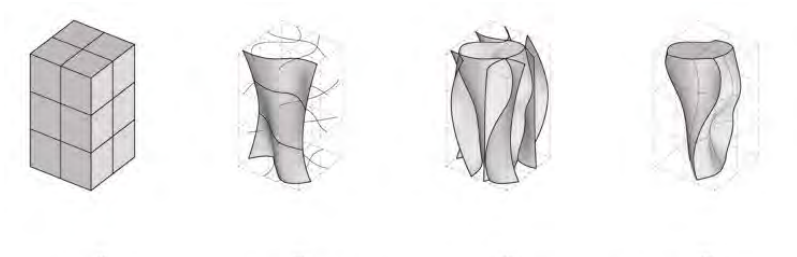
The backbone of the setup is the global parameters that are changeable within the workflow environment. Its settings, syntax and data are adapted in both the approximation plugin and the extension script while the workflow recalibrates when global settings are re-configured or need additional inputs. The global parameters allow you to toggle between "*Design*" or "*Production*" which are value bases and changes the resolution of the Elastica approximation.

One block is locally defined by its XYZ-Values (dimensions), Cut-Plane (orientation) and its local location in the XYZ-World-coordinate system and global location in a multiple-block-system.The block and blade length are interconnected to each other, if too short, the robot will move into the EPS-Block, while too large the physical implications will increase and affect the precision.



5.2 Input

A multiple block configuration is developed as framework for generation of continuous surfaces over several blocks. The "*block setup*" is framed in a Data-Tree-Structure that matches and sorts the input designs for each block and subsequently for each face of the block (6 sides). The procedure generates a data-list for each block containing cut-plane, cut-direction, number of cuts, rotation and location. By defining a clear data-flow from the input step you gain full control from design intent till export code and production.



005
Design Framework

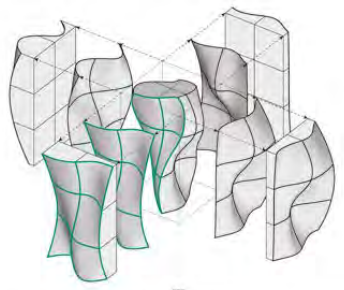
006
Geometry for cutting blocks (Design input)

007
Multiple Design "Inputs"
Data-Tree-Structure matches Design Framework

008
Brep representation of 3Dimensional design

5.3 Preprocessing of geometry

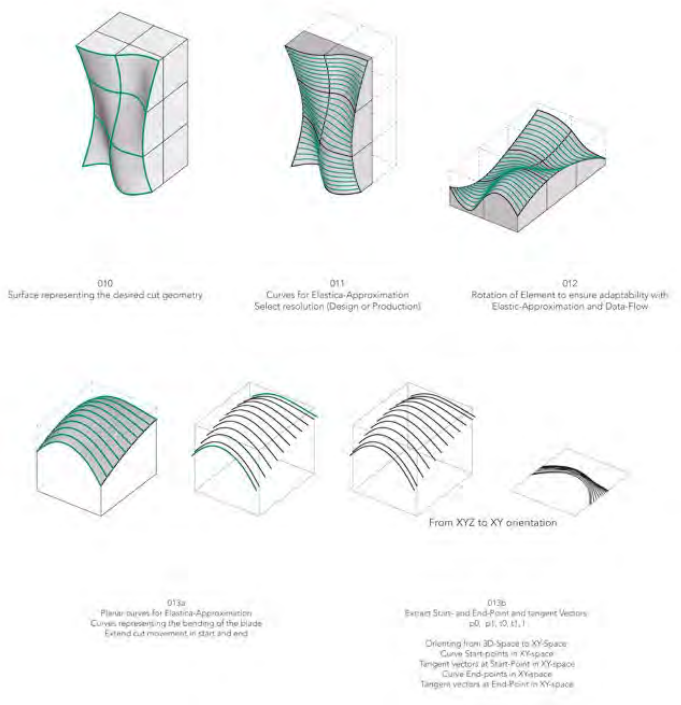
The pre-processing step first matches each block (nested in a data-tree-structure representing the design framework and block number within the design framework) with cutting geometries related to the blocks design framework. The cutting geometry is then sorted according to cut priority, primary cut being closest to the blocks base, and remaining cuts are checked for collision with the primary cut and removed if no collision occurs. The final operation generates a number of planar curves for each cut by sampling the cut geometries in the X direction of the blocks. The sampling is extended beyond the block domain to allow the robots to have lead-in and -out of each block. The number of sampling points is triggered by the current mode selected (design or production).



009
Sorting Cut-Geometries
Location and rotation of EPS-blocks in 3D space and block X and Y domain in the plane

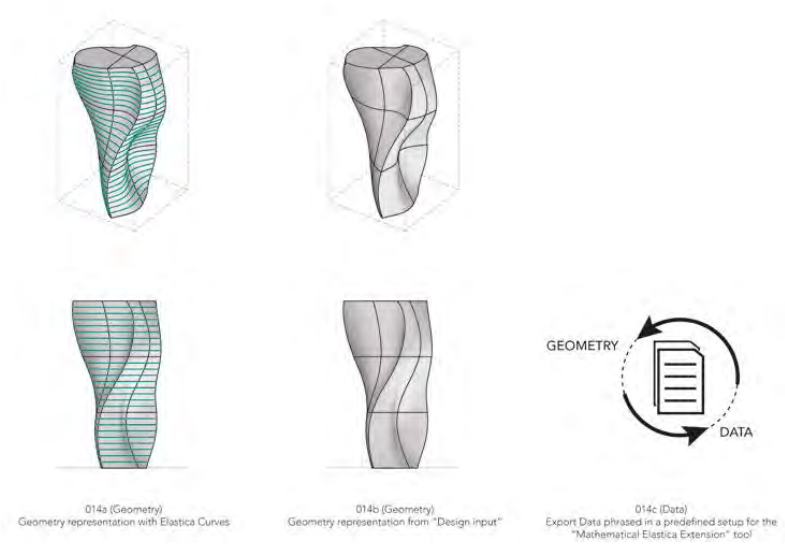
5.4 Elastica approximation

To approximate the Nurbs curves we made a Rhino plugin. The *ElasticaNum Plugin* operates on curve start-, end-points, tangent vectors at start- and end-points and the desired length of the of the elastica curve. The *ElasticaNum Plugin* is interfaced through the Rhino command-line and python code using a custom data-structure. The final operation in the *Elastica approximation* is reorienting the curves back into 3D space.



5.5 Output

The "*Design with Elastica*" tool generates two outputs that work parallel within the workflow and connect the designer with the final rationalization output while still designing in the beginning of the process. Output one is data driven exports to the "*Mathematical Elastica Extension*", while output two is representing the desired geometry by Elastica Curves. By defining a data-structure a mutual adaptation of data import and export was defined from *Input*, *Process* and *Output*. The digitized workflow outputs data from the "*Design with Elastica*" tool to the "*Mathematical Elastica Extension*" and weave the tools together. To ensure a smooth transition from one block boundary to the next, the necessity for data conversion is important to perform a mathematical extension that preserve the elastica properties.



The setup allows for a real-time *design to fabrication workflow* and a comparison between the *Design-Geometry* and the *Elastica-Geometry* are processed. While the Design-Geometry is an arbitrary input, the *"Euler Elastica Approximation"* curves are lofted with the setting on *tight*, which uses square root of chord length parameterization in the loft direction.

5.6 Design workshop

The developed toolset was subsequently tested in a workshop setting with 16 participants in the format of the *Superform: Robotic Hot-Blade Cutting* workshop, held March 15th-18th 2016 in extension to the Robarch 2016 conference at Walsh Bay, Sydney. The workshop tasked participants with formal explorations of hot-blade design potentials, produced through a dual robot setup consisting of 2x ABB IRB 1600 manipulators in a MultiMove configuration. The explorations uncovered several benefits of working directly in a production ready geometry: firstly, the exploitation of double (or more) cuts, in which two intersecting surfaces creates a sharp crease is a feature difficult to approximate through rationalization (Fig. 13, second row, middle). Secondly, the design of expressive ripple or wave-effects (fig. 14) requires careful alignment with blade-cutting direction and curvature description to remain feasible. As such, they exemplify design potentials difficult to achieve through linear rationalization.

6 Conclusion

A set of methods has been proposed for design generation of surfaces which incorporates the constraints of an elastic blade swept mechanically by two or more industrial robot manipulators. The methods are implemented as prototype design tools in C++, MatLAB, Python and GhPython

to enable interaction with non-specialist designers. The toolset was tested with 16 participants in the RobArch 2016 workshop: Superform – robotic hotblade cutting. The workshop design experiments revealed several design features that would be difficult to achieve in pure rationalization workflows, as a result of the direct incorporation of constraints and live design feedback enabled by the framework.

Figure 17: examples of the workshop participants design explorations.

7 References

Pottman et al: Architectural Geometry. Bentley Institute Press, 2007.

Brander, D., Gravesen, J., Nørbjerg, T.: Approximation by planar elastic curves. arXiv:1509.00703[math.NA].

Bruckstein, A.M., Holt, R.J., Netravali, A.N.: Discrete elastica. Applicable analysis, 78:3-4, (2001), pp. 453-485.

Bruckstein, A.M., Netravali, A.N., Richardson, T.J.: Epi-convergence of discrete elastica. Applicable Analysis, Vol. 79, (2001), pp. 137-171.

Hesse, Petra, Hg. "TailorCrete, Flight Assembled Architecture ." In Architekturteilchen. Modulares Bauen im Digitalen Zeitalter, 126-127, 164-165. Köln: 2012

Jepsen, C., Kristensen M., Kirkegaard, P: "Dynamic Double Curvature Mould System"

In: Computational Design Modeling : Proceedings of the Design Modeling Symposium, Berlin 2011 Editors: Christoph Gengnagel, Axel Kilian, Norbert Palz, Fabian Scheurer. Springer, 2011Pp. 291-300

Lim, S., Buswell, R.A., Le, T.T., Austin, S.A., Gibb, A.G.F. and Thorpe, A., “Development in construction-scale additive manufacturing processes”, Automation in Construction, Vol. 21, Issue 1, pp.262-268, 2012

Lloret Ena, Amir R. Shahab, Linus Mettler, Robert J. Flatt, Fabio Gramazio, Matthias Kohler and Silke Langenberg. "Complex concrete structures: Merging existing casting techniques with digital fabrication." Sciencedirect.com (2014):

Veenendaal, D, West, M., Block, P.: “History and overview of fabric formwork: using fabrics for concrete casting” Structural Concrete, 12, no 3. Ernst & Sohn, Berlin 2011.

Euler, L.: Methodus inveniendi lineas curvas maximi minimive proprietate gaudentes; Additamentum I: de curvis elasticis (1744).

Farin, Gerald. "A History of Curves and Surfaces in CAGD." In: *Handbook of Computer Aided Geometric Design* (2002): 1-23.

Søndergaard, A., Feringa, J., Nørbjerg, T., Steenstrup, K., Brander, D., Gravesen, G., Markvorsen, S., Bærentzen, A., Petkov, K., Hattel, J., Clausen, K., Jensen, K., Knudsen, L., Kortbek, J.: Robotic hot-blade cutting. In Robotic Fabrication in Architecture, Art and Design 2016, proceedings of RobArch 2016, (2016).

Truesdell, C. "The influence of elasticity on analysis: the classic heritage." *Bulletin of the American Mathematical Society* 9.3 (1983): 293-310.

Yamaguchi, F.: Curves and Surfaces in Computer Aided Geometric Design, Springer-Verlag Berlin Heidelberg, (1988).